UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

(2)

REPORT DOCUMENTATION PAGE

	1b. RESTRICTIVE MARKINGS
AD-A177 270 **Reproduced From Best Available Copy** ULE	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 87-0084
6a. NAME OF PERFORMING ORGANIZATION Carnegie Mellon University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research
6c. ADDRESS (City, State and ZIP Code) Robotics Institute Carnegie Mellon University Pittsburgh, PA 15213-3890	7b. ADDRESS (City, State and ZIP Code) Directorate of Mathematical & Information Sciences, Bolling AFB DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR 8b. OFFICE SYMBOL (If applicable) NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-85-0343
8c. ADDRESS (City, State and ZIP Code) Bld 410 Bolling AFB DC 20332-6448	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. 61102F; PROJECT NO. ~~2304~~ B3; TASK NO. A3; WORK UNIT NO.

11. TITLE (Include Security Classification)
CONSTRAINT-BASED SCHEDULING IN AN INTELLIGENT LOGISTICS SUPPORT SYSTEM: AN ARTIFICIAL INTELLIGENCE APPROACH

12. PERSONAL AUTHOR(S)
Mark S. Fox and ~~Stephen F. Smith~~

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final Scientific Report	FROM 3/15/82 TO 9/14/86	December 22, 1986	52

16. SUPPLEMENTARY NOTATION
(Intelligent Scheduling and Information System)

DTIC
FEB 25 1987
E

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report summarizes the research that was performed under AFOSR Contract Number ~~F49620-82-K-0017~~ AFOSR-82-0373, titled "Constraint-Based Scheduling in an Intelligent Logistics Support System: An Artificial Intelligence Approach". The goal of this research has been the development of a computational theory of constraint-directed scheduling for application to the generation and reactive management of job shop production schedules. Methodologically, the development and investigation of elements of this theory have involved the construction of a series of experimental knowledge-based systems for job shop scheduling. Several versions of a system called ISIS were developed, followed by development of a successor system called OPIS. Each of these systems were tested using simulated production data from an actual manufacturing environment. We provide an overview of this work and highlight the major accomplishments.

(Opportunistic Intelligent Scheduling).

DTIC FILE COPY

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
~~Dr. Robert Buchal~~ Hedrick	(202) 767- 4938	~~NM~~ NE

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Constraint-Based Scheduling in an Intelligent Logistics Support System: An Artificial Intelligence Approach

Final Report

AFOSR Contract Number ~~F49620-82-K-0017~~ AFOSR-85-0283

Stephen F. Smith
Mark S. Fox

Intelligent Systems Laboratory
Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

December 1986

Abstract

This report summarizes the research that was performed under AFOSR Contract Number ~~F49620-82-K-0017~~ titled "Constraint-Based Scheduling in an Intelligent Logistics Support System: An Artificial Intelligence Approach". The goal of this research has been the development of a computational theory of constraint-directed scheduling for application to the generation and reactive management of job shop production schedules. Methodologically, the development and investigation of elements of this theory have involved the construction of a series of experimental knowledge-based systems for job shop scheduling. Several versions of a system called ISIS were developed, followed by development of a successor system called OPIS. Each of these systems were tested using simulated production data from an actual manufacturing environment. We provide an overview of this work and highlight the major accomplishments.

Table of Contents

List of Figures

1. Introduction

In this report, we summarize the major accomplishments of the research performed under AFOSR Contract Number F49620-82-K 0017, titled "Constraint-Based Scheduling in an Intelligent Logistics Support System: An Artificial Intelligence Approach". The goal of this research has been the development of Artificial Intelligence (AI) based theories and techniques that enable effective computer generated solutions to real world scheduling problems. The central thesis upon which this work is based is that solutions to realistic scheduling problems require a framework that enables consideration of all relevant scheduling constraints, and, further, that knowledge about the set of relevant constraints can provide significant leverage in formulating and maintaining good solutions. Thus, our research has sought to identify these sources of knowledge, and investigate their representation and use as the basis for a constraint-directed scheduling methodology. The difficult problem of job shop scheduling was chosen as the specific focus of the research, and it is the progress made toward solution of this problem that is the subject of this report.

Given the emphasis on real-world problems, the construction of experimental knowledge-based systems for job shop scheduling has served as the primary vehicle for investigating and demonstrating our theories. Several versions of a system called ISIS (Intelligent Scheduling and Information System) have been developed, and each has been tested in the context of the Westinghouse Turbine Components Plant (WTCP) in Winston-Salem, NC using simulated plant data. ISIS makes several important contributions relating to the representation and use of constraint knowledge, and demonstrates an initial heuristic search architecture for constraint-directed scheduling. The final prototype, ISIS-II, was transferred to Westinghouse Electric Corporation in December, 1984, and its development into a production system is currently being explored under their auspices. Experience with the ISIS family of systems and an increased focus on issues of reactive scheduling has led, in turn, to the development of a successor system called OPIS (Opportunistic Intelligent Scheduler). OPIS incorporates the essential ideas contained in ISIS but introduces a dynamic, conflict-directed approach to decomposition and solution of the scheduling problem. The OPIS scheduling architecture broadens the range of constraints that can be effectively attended to and provides an integrated framework for predictive schedule generation (or expansion) and reactive schedule maintenance.

Work with these prototype systems has provided much insight into the role of constraints in scheduling, and, in the remainder of this report, we review the major accomplishments of this research. In Section 2, we discuss the nature of the job shop scheduling problem. In Sections 3 and 4, the concepts and techniques resulting from our work on the ISIS and OPIS scheduling systems

respectively are presented. Finally, in Section 5, the contributions made by this research are summarized.[1]

2. The Job Shop Scheduling Problem

Broadly speaking, the job shop scheduling problem concerns the allocation of a finite set of resources to specific manufacturing operations over time such that the orders for parts received by the factory are produced in a timely and cost-effective fashion. The production of a given order typically involves the execution of a sequence of operations, each of which possesses a specific and, for the most part, distinct set of resource requirements. Thus, in more detail, the scheduling problem consists of

- the determination of an appropriate sequence of operations, or process routing, for each order, and
- the assignment of required resources and time intervals to the operations selected.

To some extent, these two aspects of the problem are separable and, historically, this has been the case in most manufacturing organizations. Process routing selection is viewed as a planning task that is carried out during part design, and the allocation of resources to particular orders over time is viewed as the role of the scheduler. In actuality, however, there is a much greater interplay between these seemingly distinct functions. There are often several ways in which a given part can be produced (e.g., alternative machines and/or production processes may be utilized), and, while a particular routing might be designated as preferred, an a priori commitment to it ignores the dynamic nature of the actual shop floor. The feasibility of a given operation depends on the availability of its required resources, and, consequently, many process selection decisions cannot be intelligently made without consideration of the current status of the shop (e.g., current order mix, current resource levels, etc.).

The problem is further complicated by the unpredictability inherent in shop operation. Machines break down, in-process orders fail to pass intermediate quality control inspections, engineering changes are introduced, operators call in sick, and so on, all of which quickly force changes to previously planned activities. As uncertainty in the performance of activities on the shop floor increases, the usefulness of precise schedules decreases. The precision of schedules produced by a scheduler must be determined by the uncertainty of the information used in making the decisions, so as to facilitate schedule revision over time has this uncertainty is reduced. Thus, we can identify two

---

[1] Sections 4.3 and 4.4 describe research that was funded by a supplemental grant to the above mentioned contract provided by the Electronics and Material Sciences Department of the Air Force, and is included in this report for completeness in the presentation of results.

general goals in approaching the job shop scheduling problem:

- an ability to effectively *predict* shop behavior through the generation of schedules that accurately reflect the full detail of the environment and the stated objectives of the organization, and
- an ability to *reactively* revise and maintain the schedule in response to changing shop conditions.

These activities are not viewed as distinct, but rather it is felt that the same types of knowledge and methods are relevant to both. From a system engineering perspective, however, it is important to provide a system organization with the flexibility to selectively focus on specific aspects of the current schedule.

The job shop scheduling problem, in many idealized forms, is known to be NP-hard. [Garey&Johnson 79] The situation in real world scheduling environments is considerably more complex. Much of the complexity stems from the need to attend to a large and diverse set of objectives, requirements, and preferences that originate from many different sources in the plant. These scheduling influences are often in direct conflict with one another, wherein lies the crux of the problem. The production schedule must reflect a satisfactory compromise with respect to these competing influences.

2.1. Scheduling Constraints

We can partition the range of factors that influence job shop scheduling decisions into two broad classes of constraints:

- *scheduling restrictions* that serve to delineate the space of possibilities in developing a schedule, and
- *scheduling preferences* that provide a basis for differentiation amongst possible choices.

This distinction is useful to make at the outset because existing computer based scheduling systems typically give limited attention to both, and each offers distinct opportunities for improving the quality of the schedules generated.

Scheduling restrictions constrain the alternatives that may be considered in selecting and ordering operations, binding resources to operations, and designating temporal intervals during which selected operations are to take place. Collectively, they serve to define the space of admissible schedules that the scheduler must search. Scheduling restrictions include:

- *causal restrictions* - There are typically precedence constraints associated with the operations that must be performed to produce a given part, restricting the manner in which orders for that part can be routed through the job shop. A precedence constraint

on an operation states that another operation must be performed before (or after) it. These causal relationships between operations are further qualified by inter-operation travel times, transfer quantity sizes (i.e. the portion of the order that must be completed before any subsequent operations can be initiated), maximum allowable times between operations, etc. In addition, each individual operation possesses a well defined set of resource requirements which must be satisfied for specific periods of time either before or during the execution of the operation. For example, a milling operation might require the possession of a milling machine, an operator with the necessary skills, specific tools and fixtures, an appropriate NC tape, etc.

- *physical constraints* - Each machine in a job shop has specific capabilities that restrict the types of operations that can be performed on it. For example, the size of a machine's work bed may prohibit its use for operations on a particular class of parts. Likewise, each machine has particular operating characteristics (e.g., cutting speed, setup procedures) which limit the amount of work that can be performed over a certain period of time. Generally speaking, physical constraints define the functional limitations of specific resources in the shop.

- *resource unavailability* - There are also dynamic restrictions on the availability of resources that limit the scheduling alternatives available. Here we are speaking of events such as machine breakdowns that occur asynchronously and are outside of the control of the scheduler.

An understanding of the full range of scheduling alternatives is essential to the development of a realistic model of the job shop environment. The simplifications introduced in most existing computer-based scheduling systems (e.g., the designation of a single routing for each part) reduce the flexibility with which the system can respond to different scheduling problems which results in a divergence of the schedules generated from the actual real world situation. At the same time, however, any attempt to embrace the full complexity of the environment requires the ability to explicitly represent and reason with the imposed scheduling restrictions during the generation of candidate schedules.

In many problem domains addressed within the field of artificial intelligence (AI), the restrictions imposed by the problem constrain the set of admissible solutions to the extent that least commitment and constraint propagation techniques are sufficient to converge to an acceptable result [Stefik 81, Sussman&Steele 80, Waltz 75]. This is not the case in the job shop scheduling domain. Adherence to the scheduling restrictions identified above still leaves the problem severely underconstrained, and knowledge of various preferential concerns must be considered to focus the scheduler toward good solutions. These scheduling preferences fall into several categories:

- *organizational goals* - All manufacturing facilities are driven by a set of organizational goals. These goals reflect global concerns and objectives with respect to the operation of the factory, and imply general criteria against which prospective schedules can be compared. Organizational goals are established along several distinct performance dimensions. For example,

- meeting due dates - A major concern of a factory is meeting the customer due dates that are established as orders are received. The lateness of an order affects customer satisfaction and the likelihood of future business.

- minimizing work-in-process time - Work-in-process (WIP) inventory represents a substantial investment in raw materials and added value. Since these costs are not recoverable until delivery of the final product, minimizing WIP time is an important goal.

- maximizing resource utilization - Maximizing the amount of time that critical machines in the shop are actually operating (as opposed to being prepared for operation or standing idle waiting for parts) can greatly increase the overall throughput of the plant. Also, there are typically fixed costs associated with maintaining and operating the machines in a factory which can be minimized if resources are used efficiently.

- maintaining shop stability - The concern here is one of minimizing the amount of disruption to shop operations caused by revisions to the schedule. Last minute changes to the schedule can lead to increased periods of machine idle time as the preparation (or machine setup) performed in anticipation of the previously scheduled operations is undone and preparation for the newly scheduled operations is carried out.

The above performance concerns can all be viewed as approximating the overall concern of the organization: a desire to make scheduling decisions that maximize profits. They are addressed as part of the organizational planning process and lead to the establishment of specific operating expectations. For example, production levels are designated for various areas in the plant, a forecast of the number of work shifts that will be run in each area is made, and preliminary resource maintenance schedules are developed. These preferences all influence the shop schedule that is subsequently developed.

- *operational preferences* - These constraints express preferred choices amongst alternatives at the level of individual scheduling decisions (i.e., the selection of specific operations, resources, and time intervals), and reflect the heuristic knowledge present in a given scheduling environment. In many cases, these preferences provide a tactical basis for accomplishing specific global objectives. For example, an ability to effectively exploit order sequencing preferences to minimize the amount of time spent setting machines up for operations contributes directly to maximizing resource utilization. In other cases, however, such knowledge reflects an understanding of the operational characteristics of the plant that cannot be captured in predictive estimates of how well various scheduling objectives have been met. For example, a decision to avoid a particular machine when possible, based on knowledge of its susceptibility to breakdowns, cannot be properly assessed until the schedule is actually executed on the shop floor. Such decisions may in fact have detrimental effects with respect to predictive measures of schedule quality. It is important, therefore, to give proper attention to both types of concerns.

- *resource unavailability* - In contrast to the resource unavailability restrictions mentioned above, resource unavailability preferences refer to constraints that are introduced by the

scheduler. As resources are allocated to specific operations during generation or revision of the schedule, constraints declaring the resources unavailable during the allocated time periods must be generated. Such decisions must be viewed as preferences, since they may later be retracted in the face of other overriding factors (e.g. the receipt of a high priority order).

It is clear from the above discussion that an effective solution to the real world job shop scheduling problem requires an ability to reason intelligently with an amalgam of diverse constraints. Our initial discussions with plant schedulers at WTCP underscored this point. It was found that schedules were not developed in any uniform fashion, but rather through an iterative process of distributing a proposed schedule to various departments in the plant, collecting additional constraints, and attempting to alter the schedule accordingly. However, while it was clear that scheduling decisions were based on the full range of factors identified above, the lack of a methodology for balancing these concerns consistently led to generation of schedules that resulted in less than satisfactory factory performance. The schedulers were simply overburdened by the complexity of the task.

2.2. Related Research

Scheduling research to date has had relatively little impact on the real world job shop scheduling problem. Operations Management (OM) research has long studied the scheduling problem, but has done so from two, rather restrictive perspectives. The first, centered around a desire to obtain optimal results, has sought to formulate mathematical models of the problem that are tractable by linear programming techniques. This has focused attention toward simplified scheduling problems (e.g. the single machine case) which, unfortunately, have little in common with real world factory environments. A second branch of OM research has been concerned with the development of priority decision rules to provide a heuristic basis for order sequencing. These rules, while useful in making local dispatching decisions, are typically responsive to specific types of concerns (e.g. meeting due dates) and ignore all others. This restricted emphasis limits their effectiveness in more global (i.e. plant wide) decision-making contexts.

In recent years, AI research in planning has also turned attention to scheduling issues. Recognizing the limitations of reasoning with implicit notions of time, several researchers have focused on extending existing planning paradigms to include the assignment of time intervals to activities. Vere [Vere 81] describes a technique used to schedule activities aboard a spacecraft which associates time windows and durations with the various activities in the plan, and propagates refinements to this temporal information as the plan crystalizes. A similar approach is adopted in [Fukumori 80] in generating train schedules. Others have sought to reformulate the planning process within an explicit temporal framework [Allen 81, Allen&Koomen 83, McDermott 82]. Expansion of the planning problem

to explicitly address temporal concerns has also necessitated the establishment of criteria for differentiating between alternative plans. Overall plan duration has been the most common consideration in AI scheduling systems although activity cost estimates are also included in [Daniel 84] and the scheduling framework described in [Miller 83] proposes the use of special purpose critics to detect specific undesirable characteristics (e.g. deadline violations). In relating these efforts to the factory scheduling problem, the chief point of divergence is the absence of conflicting constraints. This is due in large part to the emphasis on planning the activities of a *single* agent, and the consequential lack of emphasis on efficient allocation of shared resources (i.e. resource availability is the sole concern). One exception is the NUDGE system [Goldstein&Roberts 77], which compromises between the conflicting preferences of distinct individuals in producing a schedule for a given individual's weekly activities and appointments.

3. Investigations with ISIS

Recognizing that the real world factory scheduling problem is a complex constraint-directed activity, development of the ISIS job shop scheduling system was undertaken as a means of exploring the role of constraint knowledge in generating and maintaining good production schedules. In doing so, this work has investigated issues relating to

- representing the knowledge about the factory environment and its constraints necessary to support intelligent scheduling,
- integrating constraint knowledge into the scheduling process, so as to effectively limit the generation and focus the selection of alternative scheduling decisions,
- relaxing constraints when conflict occurs, and
- using constraint knowledge to recognize and diagnose problems in the schedule.

In this section we review the research contributions that have resulted from work with ISIS. In Sections 3.1 and 3.2 we address knowledge representation issues, outlining a semantics for modeling the factory environment and its constraints. Mechanisms for interpreting the factory model and the constraint representation to generate and evaluate alternative sets of scheduling decisions are presented in Section 3.3. In Section 3.4, a hierarchical, constraint directed scheduling architecture that utilizes these mechanisms is described. Section 3.5 discusses issues of user interaction with the system. Finally, in Section 3.6, experimental results obtained with ISIS are summarized. Other accounts of this work may be found in [Fox 82, Fox 83a, Fox 83b, Fox 85, Fox 86, Fox&Smith 84a, Fox&Smith 84b, Smith 83, Smith 86a].

3.1. Modeling the Factory

One fact evident from the discussion of Section 2 is that effective scheduling decisions must reflect knowledge and constraints relating to all facets of the manufacturing enterprise. Thus, a fundamental prerequisite to scheduling is a an accurate and complete model of the production environment. The model is necessary to provide a framework for representing and organizing constraint knowledge (i.e. building the site specific knowledge base), and to impose structure that can be exploited in the development and maintenance of production schedules. However, there is a larger issue here relating to use of the model in integrating the production scheduling activity with other activities of the manufacturing organization. Our work on modeling the factory has sought to address this larger issue, and, while concentrating on the development of a representation to support constraint-directed scheduling, we have grounded this representation with a basic semantics that is applicable for modeling all aspects of the manufacturing organization.

In addressing issues of knowledge representation, we have drawn on frame-based representation techniques. In particular, our model of the factory environment and its constraints has been constructed using the Schema Representation Language (SRL) [Wright&Fox 83].[2] SRL is a frame-based language which encodes concepts as schemata. A schema is a collection of slots and values. Each schema, slot, and/or value may have meta-information attached to it. In addition to attribute knowledge, slots define inter-schema relations, along which slots and values may be inherited. The inheritance semantics of a relation are user definable. Figure 3-1 illustrates the basic SRL construct in defining an operation schema.

```
{{ operation
    {IS-A: activity
        NEXT-OPERATION: "operations which directly follow this operation"
        PREVIOUS-OPERATION: "operations which directly precede this operation"
        SUB-OPERATIONS: "operations that refine this operation at a lower level of precision"
        RESOURCE-REQUIREMENTS: "resources that must be allocated to the operation"
        ENABLED-BY: "state which enables this action"
        CAUSES: "state resulting from execution of this action"
        DURATION: "time of this action" } }}
```

Figure 3-1: Operation Schema

In this case, the description states that an **operation** IS-A type of **activity** (and, hence, inherits the

---

[2] The current OPIS system (see Section 4) is implemented in Knowledgecraft, a commercially available descendant of SRL.

attributive knowledge associated with the activity definition). This view of an operation is further refined to include the attributes (i.e. relations/slots) NEXT-OPERATION, PREVIOUS-OPERATION, etc.

To accomplish the representational goal stated above, a layered approach to developing the factory model has been adopted. Building on the basic semantics provided by SRL itself, a *world model layer* of representation is first introduced. This layer concerns the definition of general structural and relational primitives for modeling manufacturing organizations and their activities. The descriptions provided define the basic concepts of states, objects and activities, along with a set of temporal and causal relations for describing their interactions. This layer provides an epistemological framework for defining domain specific models.

The *domain layer* of the representation refines the general semantic primitives introduced in the world model layer into concepts germane to the scheduling environment. For example, in specifying the set of process routings associated with a particular product, manufacturing operations are defined as activities, and precedence constraints are composed from basic temporal and causal relations. The resources required by specific operations are expressed as objects, with their allocation represented as a collection of possession states spanning particular intervals of time. The domain-specific definitions introduced in this layer of the representation, encompassing concepts such as resources, products, product demands, operations, and materials, are themselves specialized according to important functional and structural characteristics. The resulting prototype descriptions precisely define the knowledge requirements for representing the specific entities in a particular production environment.

Within an "instantiated" factory model, the factory is represented at multiple levels of abstraction. Detailed operations are aggregated into abstract operations. Similarly, machines are aggregated by function into work areas. Thus, the factory model provides the scheduling system with multiple views of the scheduling problem, enabling the system to construct and reason about schedules at varying levels of precision.

To a large extent, the scheduling restrictions present in the factory environment, and hence the set of alternatives relevant to specific scheduling decisions, are directly reflected in the resulting model. This provides a structural framework for organizing the preferential concerns that influence various choices. This knowledge is encoded within a general constraint representation (to be discussed below in Section 3.2) and specific instances of constraints are attached directly to the relations/attributes in the model that they constrain. This is illustrated in Figure 3-2, which graphically displays a portion of a factory model centered around the description of a particular machining

operation called "P1 root grinding". Details of this framework for modeling the factory and the modeling primitives that have been defined can be found in [Fox 83a, Fox&Smith 84a, Fox 85, Sathi 85].

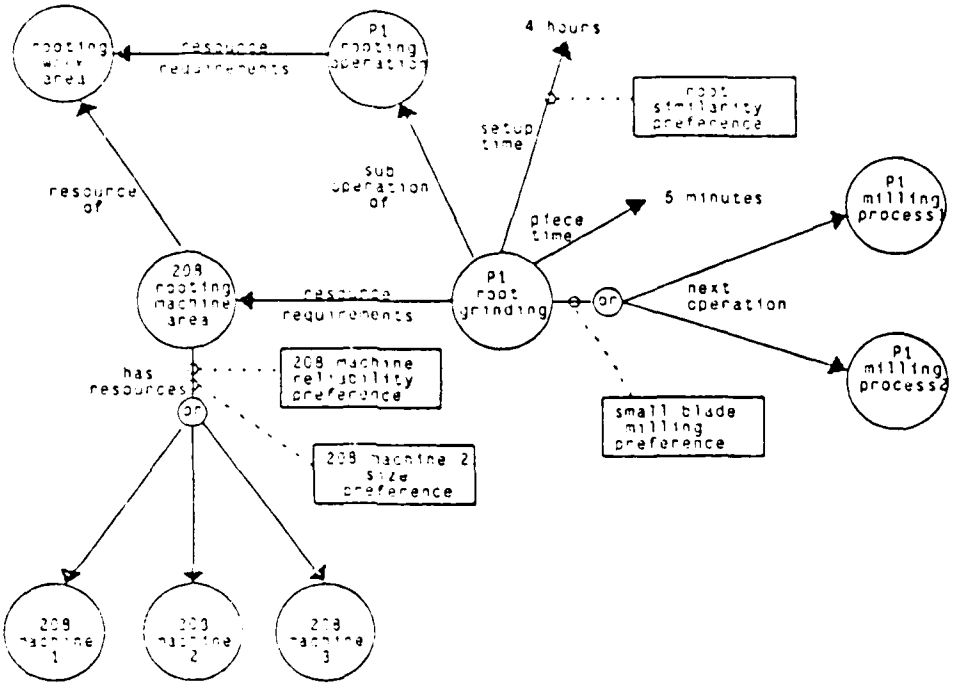


Figure 3-2: A portion of a factory model with attached constraints.

3.2. Constraint Representation

Given the central role of constraints in determining a job shop schedule, a major thrust of our research has centered on identifying and characterizing the constraint knowledge required to support an effective constraint directed search. A constraint is viewed not simply as a restriction over a set of choices, but rather as the aggregation of a variety of knowledge concerning its use. Consider the imposition of a due date. In its simplest form, this constraint would be represented by a date alone, the implication being that the job be shipped on that date. In actuality, however, due dates may not always be met, and such a representation provides no information as to how to proceed in these situations. An appropriate representation must include the additional information about the due date that may be necessary in constructing a satisfactory schedule. For example:

- what alternative dates are satisfactory if the original cannot be met?
- what preferences exist for these alternative dates?
- who specified the due date? when? and why?
- how is the satisfaction of the due date related to other constraints such as costs?
- does the satisfaction of the due date constraint positively or negatively affect the satisfaction of other constraints?
- under what circumstances should the due date constraint be considered?
- if there are two or more due date constraints specified for an order, which should be used?

Let us examine the representational issues raised by these examples, and, correspondingly, the salient features of the ISIS constraint representation. Complete details of the constraint representation may be found in [Fox 82, Fox 83a, Smith 83].

3.2.1. Alternatives and Preference Relationships

One of the central issues that must be addressed by the constraint representation is that of *conflict*. Consider capacity and due date constraints. The former may establish limits on the production capabilities of particular work areas of the plant while the latter may require shipping the order in a short period of time. Current circumstances in the shop (e.g. current shop load) may be such that accomplishment of the latter is only possible if extra work shifts are introduced, thereby causing a conflict with the former. In short, it may not be possible to satisfy both constraints, in which case one or both must be *relaxed*. This is implicitly accomplished in mathematical programming and decision theory by means of utility functions and the specifications of relaxation through bounds on a variable's value. In AI, bounds on a variable are usually specified by predicates [Stefik 81, Engleman 80] or choice sets [Sussman&Steele 80, Waltz 75].

An ability to relax a specific constraint requires knowledge of two sorts: knowledge of potential alternatives, and knowledge of the preference relationships that exist among these alternatives. Given the diversity in the types of constraints present in the job shop scheduling domain, formalization of this knowledge is accomplished by defining a taxonomy of constraint types. Constraints may be expressed either as predicates or choice sets. Choice constraints are further specialized to distinguish constraints that range over discrete or continuous choice sets. Each constraint type provides a specific framework for specifying alternatives and the preference relationships that exist among them. Expression of preference relationships is accomplished through the association of a *utility* value to each alternative, intuitively reflecting the degree to which each alternative satisfies the constraint. Utility values are defined to range from zero to one, with zero

interpreted as complete dissatisfaction and one interpreted as complete satisfaction.

As might be expected, the association of utilities to alternatives varies according to constraint type, and specific methods for deriving utility values are defined for each constraint type. In many cases, a constraint expresses preference relationships over a set of choices that are explicitly defined in the model. For example, operations are defined to take place in certain work areas of the factory, and machine preferences promote specific choices from the sets of machines in those areas. Specification of the preference relationships in these cases entails an association of specific utility values to each of the defined alternatives.

Many organizational goals, on the other hand, express preferences over a continuous and often infinite range of possible values. A due date constraint, for example, must associate a degree of satisfaction with each point along the time line. Constraints of this nature require an implicit mapping of degree of satisfaction to possible alternatives. This is accomplished by defining a characteristic function which, when evaluated with respect to a particular alternative, yields a specific utility value. This approach to specifying the preference relationships amongst alternative relaxations is not unlike the techniques employed in mathematical programming, and, in fact, allows advantage to be taken of OM heuristic priority rules that emphasize specific organizational goals.

3.2.2. Constraint Elasticity

A source of knowledge that affects decisions concerning whether or not given a constraint should be relaxed is its *elasticity*. The elasticity of a constraint is a measure relating to how "easy" it is to relax. Constraints vary in scope (i.e. the range of scheduling decisions that they constrain), and consequently vary with respect to the amount of disruption to the schedule that can be expected to result from a decision to relax the constraint. For example, a decision to relax the shift constraint associated with a particular machine (e.g. add a 2nd shift for some interval of time) affects all allocation decisions involving that resource, whereas a decision to relax (or ignore) a machine preference constraint in scheduling a particular operation only affects that particular scheduling decision. Pragmatically, this variance in the scope of different constraints translates to variance in the computational effort required to consider their relaxation. Exploration of various work shift assignments requires much more search effort than exploration that assumes preferred shift assignments. A constraint's elasticity provides guidance in determining whether the constraint should be considered relaxable and non-negotiable during the search process.

3.2.3. Constraint Importance

The relative influence to be exerted by a given constraint, i.e. its *importance*, is another type of knowledge that must enter into constraint relaxation decisions. Not all constraints are of equal importance; some are more important to satisfy than others. Specification of importance relationships within the constraint representation distinguishes between absolute and relative importance knowledge. Absolute importance knowledge relates to the static importance relationships that exist among constraints of a given type (e.g. machine preference constraints) and is specified by associating an importance metric (or weight) with each defined constraint. The importance relationships among different types of constraints are typically more dynamic in nature. For example, satisfying a due date constraint is likely to be a much more important concern than satisfying the set of relevant resource preferences in the context of a high priority order, while the opposite might be true in the case of an order generated to build inventory. Such relationships are established through the specification of *scheduling policies*[3] Scheduling policies define importance specifications which partition constraint types into distinct importance classes and associate, with each partition, a fraction of the total importance to be distributed amongst the constraints belonging to the partition. During interpretation of a scheduling policy, this fraction is distributed in proportion to the absolute importance measures associated with the constraints defined to be in that importance class.

3.2.4. Constraint Relevance

To provide a basis for determining which constraints should impact a given scheduling decision, the constraint representation also addresses knowledge relating to constraint *relevance*, the conditions under which a constraint should be applied. Given that constraints are attached directly to the schemata, slots, and/or values that they constrain in the factory model (see Section 3.1), constraint relevance can be determined to a large degree by the proximity of constraints to the portion of the model currently under consideration. For example, a given scheduling decision designates specific entities in the model (e.g. a specific order, a specific operation, a specific machine, etc.) which can be examined to collect all potentially relevant constraints. Of course there may be further conditions on the applicability of the constraints that are collected in this matter. For example, due date and WIP constraints are only relevant to decisions that make commitments with respect to the execution time of operations. Such constraint specific applicability conditions are expressed by associating a specific procedural test, termed the *context*, with each constraint specification.

There are situations in which problems arise if the applicability of constraints is based solely on their context sensitivity to the current situation. First, many constraints tend to vary over time. The number

---

[3] alternatively referred to as *scheduling goals* in [Fox 83a, Fox&Smith 84a].

of shifts, for example, fluctuates according to production levels set in the plant. Consequently, different variants of the same constraint type may be applicable during different periods of time. Within the constraint representation these situations are handled by associating a temporal scope with each variant, organizing the collection of variants according to the temporal relationships among them, and providing a resolution mechanism that exploits the organization. A second problem involves inconsistencies that might arise with respect to a given constraint type. It is possible for different variants of the same constraint type to be created and attached to the same object in the model. For example, both the material and marketing departments may place different and conflicting due date constraints on the same order. In this case, a first step has been taken in exploiting an authority model of the organization to resolve such inconsistencies.

3.2.5. Constraint Interdependencies

Another important aspect of the constraint representation concerns the *interdependencies* amongst constraints. Constraints do not exist independently of one another, but rather the satisfaction of a given constraint will typically have a positive or negative effect on the ability to satisfy other constraints. For example, removing a machine's second shift may decrease costs but may also cause an order to miss its due date. These interdependencies are expressed as relations within the constraint representation, with associated *sensitivity* and *direction* measures indicating the extent and direction of the interaction. Knowledge of these interactions can support diagnosis of the causes of unsatisfactory final solutions proposed by the system, and suggest relaxations to related constraints which may yield better results.

3.2.6. An Example

An example of a constraint within the ISIS model is a **due-date-constraint** (Figure 3-3). It constrains the range (i.e. value) that a slot may have. In particular, it constrains the DUE-DATE slot (relation) associated with an **order** schema. The predicate contained in the CONTEXT slot designates that the constraint is applicable to any decision regarding the time of an operation, and the INTERACTS-WITH relation identifies its dependency on shift constraint satisfaction decisions. The specific set of alternative values (or relaxations) designated by this constraint is described by the **due-date-constraint-spec** schema (Figure 3-4), which is defined as a type of **continuous-choice-spec**. A continuous choice spec restricts the value of a slot to a particular domain, in this case the domain of dates, and specifies a piece-wise linear utility function over this domain. This function provides the basis for determining the utility of any particular value chosen (via interpolation).

The **fo-due-date-constraint** schema, depicted in Figure 3-5, defines the due date constraint

```
{{ due-date-constraint
    {IS-A range-constraint
        IMPORTANCE:
        CONTEXT: time-commitment-madep
        INTERACTS-WITH: shift-constraint
            direction: negative
        DOMAIN:
            range: (type IS-A order)
        RELATION: due-date
        CONSTRAINED-BY:
            range: (type IS-A due-date-constraint-spec) }
    PRIORITY-CLASS:   }}
```

Figure 3-3: due-date-constraint schema

```
{{ due-date-constraint-spec
    { IS-A continuous-choice-spec
        DOMAIN: dates
        PIECE-WISE-LINEAR-UTILITY:
        EVALUATOR: interpolate }  }}
```

Figure 3-4: due-date-constraint-spec schema

associated with "forced outage" orders. The utility function is specified by <shipping-lateness utility> pairs, and states that the utility of a particular choice will be

- 1 if the due date chosen is on or before the requested due date,
- linearly decreasing from 1 to 0.1 if the due date chosen is between 0 and 7 days late, and
- 0.1 if the due date chosen is more than 7 days late.

3.3. Interpreting Constraints to Generate and Evaluate Alternatives

We have already observed that exploration of alternatives (i.e. search) is an integral part of generating constraint-satisfying schedules. Indeed, the constraint representation summarized in the previous section emphasizes the knowledge required to support an effective constraint directed search. In this section, we describe basic mechanisms for interpreting the constraint representation to generate and evaluate search alternatives. Later, in Section 3.4, we examine how these mechanisms are exploited within the ISIS search architecture.

```
{{fo-due-date-constraint
    {IS-A: due-date-constraint
        CONSTRAINED-BY: {{INSTANCE due-date-constraint-spec
                                PIECE-WISE-LINEAR-UTILITY: ((0 1.0) (7 0.1)) }} }
    PRIORITY-CLASS: forced-outage }}
```

Figure 3-5: fo-due-date-constraint schema

For the most part, the alternatives we are speaking of are partial solutions - subsets of the total set of scheduling decisions that comprise the factory schedule. Candidate partial solutions under consideration might, for example, represent alternative sets of decisions that could be taken with respect to a particular order. This emphasis on the exploration of alternative partial solutions is due to the combinatorics of the underlying search space, which makes it necessary to focus incrementally on specific aspects of the schedule and make commitments prior to seeing a complete schedule. We will use the terms hypothesis, solution component, and partial schedule interchangeably to refer to a candidate partial solution.

3.3.1. Constraint Resolution and Constraint-Based Evaluation

Central to an ability to exploit constraint knowledge in the evaluation of alternatives is a means for constraint resolution: the determination of precisely which constraints should impact the scheduling decision (or decisions) under consideration. Let us assume (for the moment) the case where a hypothesis consists of a single scheduling decision. In this case, knowledge of where constraints are placed in the factory model (see Section 3.1) can be used in conjunction with the relevance knowledge that is associated with individual constraints (see Section 3.2) to define a general resolution mechanism:

- The appropriate slots of the schemata representing each entity identified in the prospective scheduling decision are scanned to collect all potentially relevant constraints.
- this set is then filtered through application of constraint-specific applicability conditions and resolution mechanisms to yield the final set of constraints.

Extending the constraint resolution mechanism to handle hypotheses representing a set of scheduling decisions is only slightly more complex. There is one additional concern: In applying the above resolution mechanism to each decision represented by the hypothesis we may retrieve the same constraint more than once. For example, if the hypothesis proposes execution times for more

than one operation for a given order, then the order's due date constraint will be resolved to be relevant to both decisions. However, the constraint should only be applied once. This is handled by classifying constraints into two categories: *invariant* and *transient*. Invariant constraints are always retained, whereas only the most appropriate variant of a transient constraint is retained. In the case of the due date constraint (which involves a prediction of when the order will finish), the appropriate variant would be the one associated with the operation that is furthest downstream in the order's production plan.[4]

Through the use of assigned utilities that express a given constraint's preferences relative to possible choices (or relaxations), and knowledge of the importance relationships that exist among constraints, a mechanism for evaluating alternative hypotheses is defined. The rating scheme intuitively reflects how well a given hypothesis satisfies the relevant constraints. This is accomplished as follows:

- All constraints relevant to the hypothesis (as determined by the constraint resolution mechanism described above) are applied. Each participating constraint imparts a utility in the range from zero to one reflecting how well it is satisfied by the hypothesis.

- The appropriate scheduling policy is applied to determine the influence (or weight) that each constraint should be given in the evaluation. As mentioned in Section 3.2, the scheduling policy defines a partition of constraint types according to importance classes, and ascribes a specific percentage of importance to each partition. The percentage of importance allocated to each importance class is distributed amongst the constraints belonging to that partition in proportion to the absolute importance measures associated with each constraint. This results in the assignment of a weight to each participating constraint, where each weight is a value between zero and one and the total set of weights sums to one.

- The rating assigned to the hypothesis is the weighted sum of the utilities assigned by the participating constraints.

This evaluation scheme provides a framework for selective evaluation of the evolving solution at different levels of aggregation. For example, a subtask concerned with the generation of a particular solution component (e.g. decisions relating to a particular order) can be driven by constraint satisfaction assessments local to that portion of the solution. As sets of scheduling decisions are

---

[4] If assumptions can be made about the manner in which hypotheses are generated, then the above procedure for constraint resolution can be optimized without loss of generality. For example, ISIS generates hypotheses in an incremental fashion (i.e. at each step of the search, one or more existing hypotheses are extended to include an additional scheduling decision). Because of this, constraint resolution can be, and is, implemented as an incremental process of determining the constraints that are relevant to the hypothesis extension, and "merging" these constraints with those were determined to be relevant during prior development of the hypothesis. Additional advantage is taken from the fact that ISIS works either forward or backward through an order's production plan when considering alternative schedules. Given this fact, the most recent variant of a transient constraint is always the one that should be retained.

combined to form more encompassing partial solutions (e.g. as schedules for individual orders are integrated into a shop schedule), corresponding aggregate measures of constraint satisfaction can be produced by appropriately merging previously collected sets of constraints. The scheme also provides a means for measuring the extent to which specific influences have been attended to at different levels of aggregation. With respect to meeting deadlines, for example, measures reflecting "how well order 10's due date constraint is satisfied", "how well the end time constraints in the milling work-area are satisfied", "how well the due date constraints of orders of priority class x are satisfied", are all obtainable by focusing on different cross-sections of the previously determined set of relevant constraints. These distinct levels of evaluation can all provide useful information upon which to base search control decisions.

3.3.2. Constraint-Based Generation and Constraint Relaxation

The restrictions present in the factory model can be interpreted as a set of basic search operators and, hence, provide a basis for generating alternative hypotheses. For example, a resource requirement constraint stating that the P1-root-grinding must take place in the 208-rooting-machine-area (see Figure 3-2) provides a basis for generating 3 competing hypotheses: one that represents the selection of each machine residing in that work area of the plant to perform the P1-root-grinding operation. Similarly, the operation precedence constraints depicted in Figure 3-2 that specify P1-milling-process1 and P1-milling-process2 as alternative successors to the P1-rooting-operation provide a basis for extending each of these three hypotheses in two different ways, each designating the selection of a particular successor operation. Duration and capacity constraints provide the basis for extending hypotheses to include alternative operation execution times.

Use of such search operators in an unconstrained fashion, however, results in a combinatorial explosion of alternatives, and knowledge must be exploited to limit the search. In some cases, the basic search operators can be specialized by exploiting the preference constraints that influence the alternatives they generate. Going back to Figure 3-2 once more, suppose that the constraint specified by **small-blade-milling-preference** defines a strong preference for P1-milling-process1 (i.e. it has a high importance value) and that the order for which scheduling decisions are being generated concerns the production of small blades (i.e. the constraint's condition of applicability evaluates to true). This knowledge can be used to restrict the generation of alternatives, in this case removing the P1-milling-process2 operation from consideration.

In the absence of strong preferential guidance, the generation of alternatives can be constrained by placing limits on the number of hypotheses that may be extended at any step of the search. The beam search employed by ISIS (see Section 3.4 below) operates in this fashion, using the constraint-based

evaluation scheme described above to determine which hypotheses to retain and extend.

This constraint-based hypothesis generation paradigm defines a *generative* constraint relaxation process. The application of a given search operator generates alternative scheduling decisions, each of which specifies an alternative relaxation of the constraint (or constraints) from which the search operator was derived. Furthermore, each alternative generated makes implicit relaxation decisions with respect to all other constraints relevant to the scheduling decision. Thus, as the search proceeds, solutions which relax various constraints to various degrees are considered.

Of course, to make generative relaxation feasible, it must operate in tandem with *analytic* constraint relaxation processes. Given the size and complexity of the search space, it is necessary to make explicit decisions that bound the dimensions of the search to be conducted (e.g. which set of scheduling decisions to consider next, which search operators to use). Similarly, the fact that the search is being heuristically restricted requires an ability to redirect the search (i.e. alter its dimensions) upon recognition that the search has produced an unsatisfactory compromise. Each of these search control decisions ultimately influences, in one way or another, the degree to which various constraints will be relaxed. The extent to which this decision-making can be totally automated remains an open question, and within ISIS, many of these control decisions are predefined by the overall system architecture. At the same time, it has been possible to define rule-based relaxation processes, driven in part by the constraint elasticity and interdependency knowledge described in Section 3.2 and in part by characteristics of the constraint conflicts that must be resolved, that address some of these control issues. The ISIS architecture utilizes rule-based components for initializing local searches and diagnosing some specific constraint satisfaction failures (see Section 3.4.3 and [Fox 83a]). Within the OPIS scheduling architecture (which actually provides a more appropriate framework for diagnostic processes), these ideas are extended to encompass strategic decisions relating to problem decomposition (see Section 4).

3.4. Hierarchical, Constraint-Directed Scheduling

The basic mechanisms described in the last section form the nucleus of the ISIS approach to generating and maintaining job shop schedules. Both these mechanisms and an overall framework for applying them have evolved over the course of this research, resulting in several distinct versions of ISIS. The initial ISIS search architecture employed these mechanisms in a strictly non-hierarchical fashion. Experimentation with this version of the system pointed out a susceptibility to "horizon effect" problems, and additional levels of analysis were added in subsequent versions to combat these problems. In this section we describe the operation of the final hierarchical version of ISIS, and compare its performance characteristics to the earlier, non-hierarchical version. See [Fox 86] for a

retrospective look at the evolution of this search architecture.

In constructing a job shop schedule, the ISIS search architecture assumes an "order-based" scheduling perspective. By this we mean that an initial decomposition of the problem is performed wherein the orders that require scheduling are prioritized. The shop schedule is then derived by incrementally determining a schedule for each individual order. Thus, the complexity of the overall scheduling problem is reduced by restricting system attention, at any point, to the decisions surrounding a particular order. The generation of a given order's schedule is cast as a hierarchical, constraint-directed search. Different levels of the search operate with different abstractions of the problem, each a function of the types of constraints that are considered at that level. Control generally flows in a top down fashion, moving through successively more detailed levels of analysis. This is illustrated in Figure 3-6. The following subsections summarize the major components of this search architecture.

3.4.1. Problem Decomposition

The order selection level of analysis constitutes the system's global problem decomposition strategy. It collects the set of orders that require scheduling (e.g. newly released orders, partially scheduled orders, previously scheduled orders whose schedules have been affected by unanticipated events and/or decisions imposed by the user), and assigns a priority to each. This prioritization of the orders to be scheduled provides a high level, order-by-order plan for completing the shop schedule. The ISIS search manager carries out this plan by selecting orders for scheduling in priority order. The scheduling of a given order may disrupt existing schedules for lower priority orders, in which case the affected lower priority orders are queued for rescheduling. The particular prioritization schemes that have been tested presume a grouping of orders into distinct priority classes, and base the order priority calculation on both priority class and the closeness of the requested due date.

3.4.2. Constructing an Order's Schedule

Construction of the shop schedule proceeds by repeatedly selecting and scheduling the unscheduled order with the highest priority. We found that order-centered search in the presence of bottlenecked resources resulted in a significant horizon effect. The need became obvious for a hierarchical search in which an abstracted version of the problem, focusing on critical resource capacity, was solved. Consequently, a capacity-centered level of analysis is first applied to propagate the temporal consequences of the requested start and due dates assigned to the selected order. The result is a coarse schedule that reflects due date considerations in the context of the current shop load. Propagation is carried out by means of a dynamic programming analysis that elaborates the set

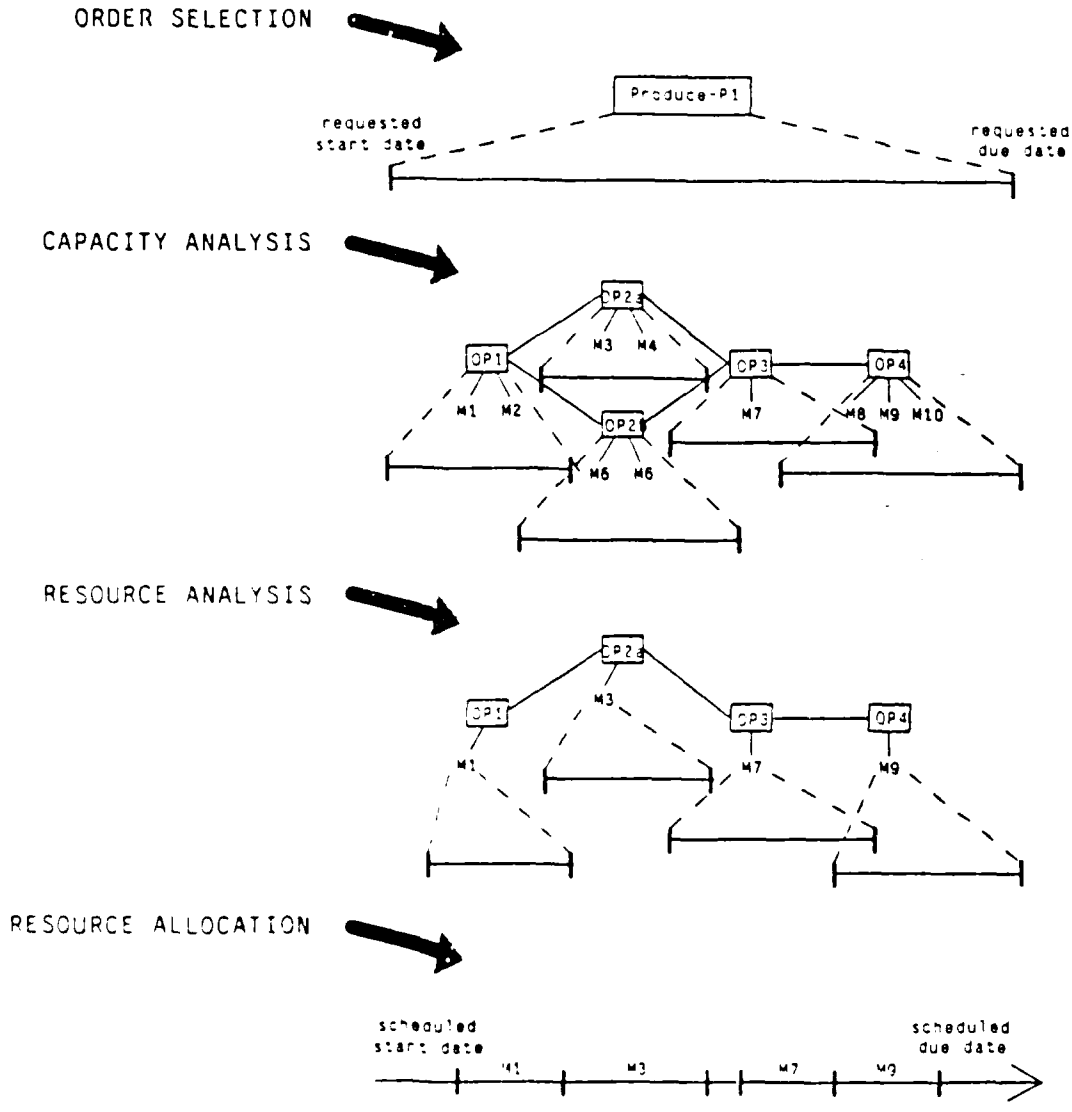


Figure 3-6: Successive refinement of an order's schedule within ISIS

of possible routings for the selected order, and associates an earliest start time and latest finish time with each operation. This information is embodied in a set of preference constraints which serve to influence the decisions that will be made during the subsequent detailed resource analysis. If these constraints are satisfied by the final schedule produced, then the order will be completed within its externally imposed release and due dates. They are cast as preference constraints, however, to

enable compromise with respect to other conflicting concerns.

The detailed resource analysis level considers the full range of restrictive and preferential constraints that surround the production of the current order, and it is at this level that the constraint-based generation and evaluation mechanisms described in Section 3.3 come into play. Again operating over the set of possible routings, a heuristic search is performed that proceeds either forward from the order's requested start date or backward from its requested due date. Alternative schedules for the order are explored incrementally - on each iteration of the search the current set of candidate partial schedules is expanded by considering one additional scheduling decision (e.g. the selection of an operation to perform, the selection of a resource for an operation, the selection of a time interval for an operation[5]). Using a beam search, only the *n* best partial schedules are retained and extended at each iteration. Constraints are collected and applied as described in Section 3.3.1 to assess how well each candidate satisfies relevant preferences and this provides the basis for pruning. Upon completion of the search, a commitment is made to the highest rated hypothesis. Constraints reflecting these decisions are posted to restrict the final determination of the order's schedule. The inclusion of the operation time bound constraints from the capacity level has what we call a *periscope* effect on the search. Their consideration in the local evaluation of a partial schedule provides a look ahead into the possible consequences of the decisions.

The schedule produced during detailed resource analysis significantly refines the coarse schedule generated at the capacity analysis level of the search. A specific process routing has been selected for the order under consideration, resources have been selected for each operation in that routing, and resource time bound constraints have been associated with each selected resource. Complete specification of the order's schedule at this stage requires only the refinement of the imposed resource time bounds. The resource assignment level of the search carries out this refinement, leading to final allocation decisions for each resource required in the order's schedule. Operating within the time bounds imposed by detailed resource analysis, allocation decisions are made that attempt to minimize the order's WIP time. Once finalized, these decisions are added to the existing shop schedule and serve to further constrain any subsequent scheduling that is required.

---

[5] A variety of alternatives exist for each type of operator. For example, two operators have been tested for choosing the execution time of an operation (see Section 3.6). The "eager reserver" operator chooses the earliest possible reservation for the operation's required resources, and the "wait and see" operator tentatively reserves as much time as available, leaving the final decision to the resource assignment level of analysis.

3.4.3. Deviating from the High Level Plan

This top down approach to generating an order's schedule constitutes the system's default order scheduling plan. The ISIS search architecture provides a framework for deviating from this plan in problematic situations by associating pre-search and post-search analysis phases with each level of the search. Post-search analysis is concerned with detection of unacceptable search results (i.e. poorly satisfied constraints) and identification of prior decisions that are likely to have caused the problem. If problems are encountered, the diagnosis identifies the appropriate level at which to redirect the search. Pre-search analysis responds to diagnosed problems by altering the set of assumptions under which the targeted level of the search will proceed (i.e. relaxing specific constraints that would otherwise be considered non-negotiable). In practice, these aspects of the ISIS search architecture have not been extensively explored. This has been due primarily to difficulties in mapping appropriate prescriptive actions into the specific levels of analysis conducted by ISIS. More generally, it has been recognized that the system architecture's commitment to an order-based scheduling perspective confounds reaction to many types of problems. The OPIS scheduling architecture (see Section 4) alleviates this problem and provides a more appropriate framework for reactive control.

3.4.4. Dealing with Shop Floor Plan Deviations

A rudimentary facility was provided for handling problems on the shop floor which forced deviations from plan, (e.g., machine breakdowns). Since a schedule is viewed as a set of constraints on the availability of resources, deviations were viewed as constraint violations. Our approach implemented a policy that the repaired schedule deviate as little as possible from the original in order to reduce shop instability. This was accomplished by turning the original schedule's resource reservation constraints into preference constraints, to be used in the rescheduling of the affected orders.

3.5. Interaction with the User

The ISIS user interface is viewed as a medium for communicating constraints to the system. The user specifies what the constraints are, and the schedules produced are responsive to these concerns. To facilitate acquisition and refinement of this constraint knowledge, a number of high level interfaces are provided. The constraint editor is used to formulate preference constraints. Driven by knowledge of the underlying constraint representation, it provides guidance to the user in specifying or revising the necessary information relating to relevance, importance, and partial satisfaction. Once specified, a new constraint is automatically integrated into the existing knowledge base. Similar editors are provided to facilitate changes to other aspects of the factory model. A status update interface is used to communicate new scheduling restrictions that result from factory operation.

The interactive scheduling subsystem provides a graphical interface through which the user can manually perform some portion of the scheduling task. The user may elect to make specific scheduling decisions prior to involving the automatic scheduler (e.g. manually schedule a critical area of the plant), or manually adjust the automatically generated schedule after the fact. Scheduling decisions imposed by the user are treated as additional constraints during subsequent automatic scheduling.

As individual scheduling decisions are made by the user, the system uses its constraint knowledge in an advisory capacity. Relevant scheduling restrictions are checked for constraint violations (e.g. the operation being scheduled cannot be performed on the machine indicated), and, if any are found, feasible alternatives are suggested. If a proposed scheduling decision is found to satisfy all scheduling restrictions, the decision is evaluated with respect to relevant preferential concerns. Once again, constraints are collected and applied in the manner described in Section 3.3.1, and the satisfaction estimates returned are used to provide the user with an indication of the desirability of the decision. A sample commentary is shown in Figure 3-7. In this case, five distinct preference constraints were found to be relevant to the decision in question. The partitioning of these considerations reflects the particular scheduling policy associated with the order being scheduled. These assessments make the user aware of all constraints that the system knows to be relevant to specific scheduling decisions. In doing so, they also provide a context for identifying constraints that are incorrectly specified or currently unknown to ISIS.

3.6. Experimental Results

As indicated at the outset of this report, the work on ISIS was carried out in the context of the Westinghouse Turbine Components Plant (WTCP) in Winston-Salem, NC. The WTCP scheduling problem addressed during the ISIS development effort was restricted to the portion of the plant responsible for the production of steam turbine blades, which constituted approximately one third of the total shop floor area. A turbine blade is a complex three dimensional object produced by a sequence of forging, milling and grinding operations to tolerances of a thousandth of an inch. Thousands of different blade styles are produced in the plant, primarily in batches of 1 to 200 blades. Orders released to the floor fall into distinct priority classes which range from replacement orders for malfunctioning blades in currently operating turbines, to blade orders that accompany orders for new turbines, to orders for blades to be placed in stock for future use. There are typically 100 to 200 blade orders on the shop floor at any time.

Each style of turbine blade produced in the plant has one or more possible process routings associated with it, each ranging in length from 10 to 15 operations. Distinctions between alternative

Decision being contemplated:

order: mo-00039
resource: r208-5
operation: op.4-CSE1
start-time: Wed Apr 10 1985
end time: Fri Apr 12 1985

Primary considerations {Importance >= 30%}:

Sufficient lead time exists to complete preceding operations on order mo-00039 if started by the requested start date of Tue Apr 2, 1985.

Due date constraint sufficiently satisfied. Order mo-00039 should finish early by 4 day[s] 4 hour[s].

Secondary considerations {Importance < 30%}:

r208-5 was a preferred choice because number-of-lugs of product was satisfied.

The preceding order on r208-5 is not of the same airfoil-type. No sequencing advantage taken.

The following order on r208-5 is of the same airfoil-type. Good sequencing decision.

Figure 3-7: Evaluating a Scheduling Decision

routings may be as simple as substituting a different machine, or as complex as changing the manufacturing process. In-process orders in the shop must share the use of approximately 50 machines and human manned work centers, as well as a full array of supporting resources (e.g. operators, tooling, nc tapes, box gauges, etc.). Shop floor scheduling at WTCP is a formidable task, and decisions are influenced by the full range of concerns outlined in Section 2.

Experiments have been conducted relative to the WTCP scheduling problem with several versions of the ISIS scheduling system. In each experiment, an empty job shop was loaded with a representative set of 85 blade orders spanning a period of two years. The various types of constraint knowledge influencing the development of schedules in these experiments included:

- alternative operations,
- alternative machines,
- due dates,

- start dates,
- operation time bounds,
- order priority classification (with orders falling into 4 priority classes),
- work in process restrictions,
- sequencing constraints to reduce setup time,
- machine constraints on product form and length,
- resource availability, and
- shop stability (minimizing pre-emption).

A total of 13 experiments were performed. These experiments have explored the effects of alternative constraints, alternative search operators, and the utility of the hierarchical search architecture. In this section we will examine the results obtained in two selected experiments, which serve to underscore the advantages of the hierarchical search architecture. A detailed discussion of all experiments may be found in [Fox 83a].

To provide a benchmark for comparison, the initial version of ISIS tested was non-hierarchical, employing only the detailed (beam search) level of scheduling. Assignment of reservation times in this experiment was handled by the eager reserver. The gantt chart[6] shown in Figure 3-8 depicts the schedule that was generated by this version of the system. The schedule is a poor one; 65 of the 85 orders scheduled were tardy. To compound the problem, order tardiness led to high work-in-process times (an average of 305.15 days) with an overall makespan[7] of 857.4 days. The reason for these results stems from the inability of the beam search to anticipate the bottleneck in the "final straightening area" of the plant (the fts* machine on the gantt chart in Figure 3-8) during the early stages of its search. Had the bottleneck operation been known in advance, orders could have been started closer to the time they were received by the plant and scheduled earlier through the bottleneck operation.

The version of ISIS producing the best results in these experiments was the hierarchical system

---

[6] Each row represents a machine, and each column a week. If a position in the gantt chart is empty, then the machine is idle for that week. If a position contains an "o", then it is utilized for less than 50% of of its capacity. If the position contains a "@", then over 50% of its capacity is utilized. Machines that are encountered earlier in the process routings appear closer to the top of the chart.

[7] Makespan is the time taken to complete all orders.

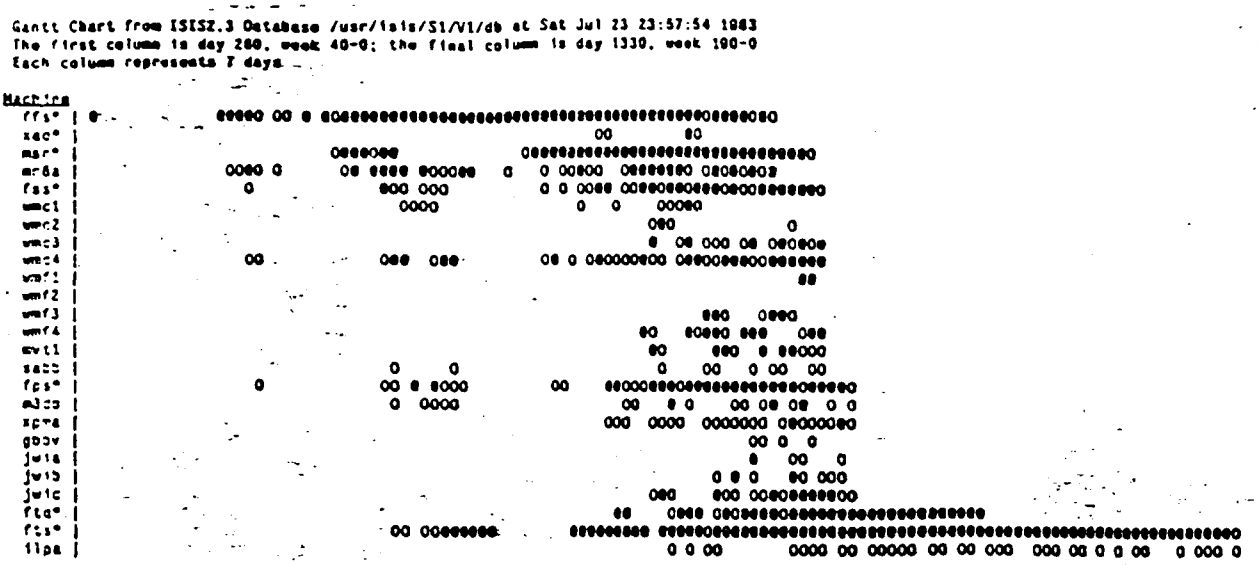


Figure 3-8: Version 1 Gantt Chart

described in Section3.4 employing the wait-and-see reserver. The schedule generated in this experiment is shown in Figure 3-9.

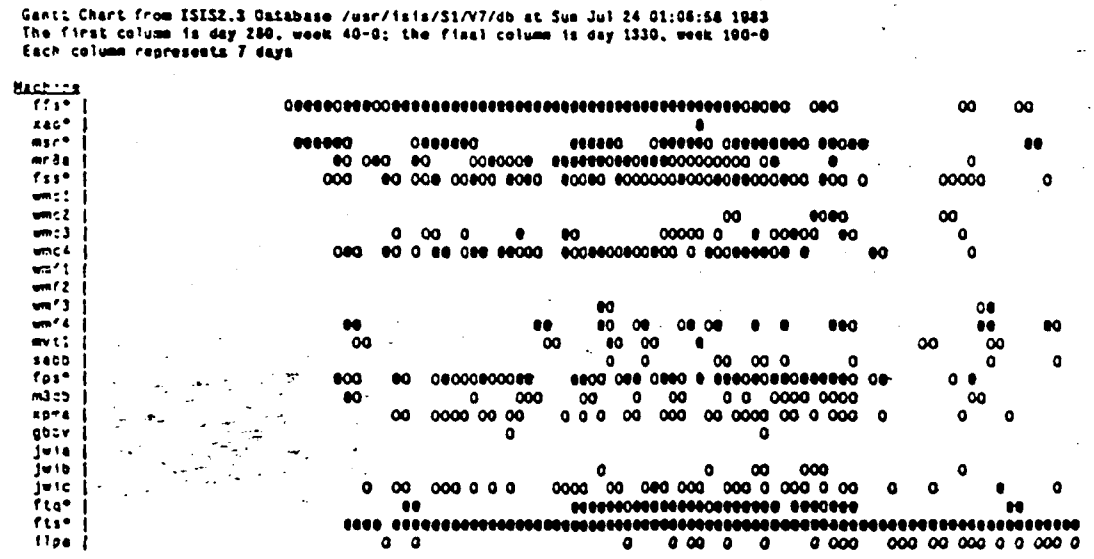


Figure 3-9: Version 7 Gantt Chart

The abstract view of the problem provided by the capacity-based level of scheduling led to a considerable improvement in performance, evidenced most dramatically in the increased satisfaction

of the due date constraints. The average utility assigned by the due date constraint to lower priority "service orders", for example, almost doubled, rising from a value of 0.46 in the first experiment to a value of 0.80.[8] The total number of tardy orders was reduced to 14. Moreover, a much lower average work-in-process time of 186.73 days was achieved, resulting in an overall makespan of 583.25 days. In this case, inadequate machine capacity in the "final straightening area" (fts*) appeared to be the principal limitation affecting order tardiness.

4. Investigations with OPIS

As experience was accumulated with the ISIS scheduling architecture, weaknesses stemming from its strict reliance on an order-based decomposition of the problem were perceived. It was recognized that an a priori commitment to a single "scheduling perspective" introduced a bias with respect to the types of constraint conflicts that could be effectively resolved, in this case resulting in poor satisfaction of constraints surrounding the allocation of specific resources (e.g. sequencing preferences to minimize machine setup changes) [Smith&Ow 85]. To effectively attend to the full range of constraints, it must be possible to selectively adopt different scheduling perspectives.

More generally, these problems reflect a need to reason about constraints and constraint relaxation at higher levels. The research conducted with ISIS emphasized mechanisms for constraint-directed reasoning at the "micro" (or individual decision) level (see Section 3.3). However, many global search control decisions (such as those relating to how the problem should be decomposed) affect the system's ability to satisfy classes of constraints, and hence must be based on more "macro" level analyses of the problem constraints; in particular an ability to recognize important constraint conflicts (or types of conflicts) and direct problem solving activity accordingly is essential to effective use of "micro" level constraint relaxation techniques. The OPIS scheduling system grew out of a recognition of these problems, and the desire to investigate the potential benefits of a dynamic, conflict-directed approach to problem decomposition.

In this section, we highlight the major ideas that have emerged from work with OPIS. In Section 4.1, we consider the implications of different scheduling perspectives with respect to constraint satisfaction. An initial multi-perspective scheduling system, termed OPIS 0, is presented in Section 4.2, and the results of a comparative analysis of multi-perspective and single perspective scheduling are summarized. In Section 4.3, a generalized framework for conflict-directed problem decomposition is described. Finally, in Section 4.4, the stock of scheduling methods currently implemented in OPIS,

---

[8] *These values were obtained using an earlier version of the constraint evaluation scheme described in Section 3.3.1 that assumed a rating scale from 0 (totally unsatisfied) to 2 (completely satisfied).*

and the scheduling strategies they give rise to, are summarized. Various aspects of this work are discussed in [LePape&Smith 86, Ow 86, Ow&Smith 86a, Ow&Smith 86b, Ow&Smith 86c, Smith&Ow 85, Smith 86b, Smith 86a, Smith 86c].

4.1. The Case for Multiple Scheduling Perspectives

As indicated above, different decompositions of a scheduling problem may be obtained by adopting different scheduling perspectives. These perspectives provide high level alternatives that a scheduling system might consider. In this section we discuss two such scheduling perspectives:

- a *resource-based perspective*, where the shop schedule is viewed as a collection of resource schedules (i.e. work area and/or machine schedules), and decomposition of the scheduling problem centers around the development of schedules for individual resources, and
- an *order-based perspective*, where the shop schedule is viewed as a collection of order schedules, and decomposition of the scheduling problem centers around the development of schedules for individual orders.

Each of these scheduling perspectives advocates a specific local and incomplete view of the overall scheduling problem in terms of more tractable subproblems. As such, a decision to employ any one perspective has its advantages and disadvantages with respect to the solution constraints that have been imposed, and neither perspective dominates the other. In order to determine where each perspective can be most effectively used, and, hence provide a basis for selective use of both, it is necessary to understand the implications of each decomposition perspective so as to identify its strengths and weaknesses.

An order-based perspective is the one adopted within ISIS. At each step of the generation of a shop schedule, a specific order is selected and a schedule is developed for it, taking into account only the resource allocation decisions that have been made with respect to previously scheduled orders. The implicit assumption here is that the decisions to be taken for orders that are still unscheduled are less important (i.e. it is possible to accept a lower level of constraint satisfaction for them without significantly disturbing the quality of the overall schedule). A decomposition based on this perspective results in subproblems that group together the constraints surrounding a particular order to be scheduled, and thus provides an opportunity for effectively resolving order-centered constraint conflicts. These conflicts involve only the constraints associated with a particular order, and information about other orders is not required to reason about them. An order-centered constraint conflict might, for example, involve an order's operation precedence constraints and its due date

constraint[9]. Alternatively a conflict might exist between a work-in-process constraint that tries to limit the length of time an order remains in the shop, and a constraint that expresses a preference for particular resources to be used for an operation. An order based decomposition allows the direct exploration of alternative decisions so as to determine the best compromises amongst conflicting order centered constraints.

Order-centered conflicts may be contrasted with resource-centered constraint conflicts, which arise from the need to share resources, and involve constraints associated with several orders. Limited capacity in the shop often results in competition between orders for certain resources over some period of time. Competition for a resource is typically driven by time constraints placed on the use of the resource, e.g. a constraint that an operation must finish by a certain time to avoid penalties. Yet, such competition can often be lessened (or completely resolved) if proper attention is given to resource-centered constraints. For example, by properly exploiting order sequencing preferences to minimize the amount of setup time at a particular resource, it might be possible to process all orders that require the resource within the time constraints imposed. It is evident that an order-based decomposition is inappropriate for handling such conflicts as it disperses the constraints involved in resource-centered conflicts across a number of subproblems. Given an order-based decomposition, the extent of a resource-centered conflict can only be partially determined by evaluating how the constraints associated with the current order are affected by previously scheduled orders. The constraints associated with orders that have yet to be scheduled are not available for consideration. Similarly, exploration of alternative ways of resolving conflicts is limited to the alternative ways that the current order can be scheduled in the context of the partially developed shop schedule. Possible synergies with the scheduling that remains to be done must be completely overlooked.

A problem decomposition based on a resource-based perspective, alternatively, provides a direct means for addressing resource-centered conflicts. In this case, the constraints surrounding the use of particular resources are grouped together in the subproblems that result from the problem decomposition. These constraints cut across all orders that may require the resources. Such a perspective promotes both the detection of resource-centered conflicts and the evaluation of the extent of such conflicts, since information about the relevant constraints of all orders involved is available. Furthermore, since the scheduling decisions concern more than one order, there is greater flexibility in exploring alternative ways for resolving these conflicts. It is possible to consider different sequences of orders, for example, based on the sequencing preferences that are present.

---

[9] e.g. if there had not been a precedence constraint on operations, it might have been possible to perform certain operations in parallel so that an order may complete on time

Nonetheless, just as the disadvantages of an order-based decomposition arose from its resulting grouping of constraints, so do the disadvantages of a resource-based decomposition. In this case the grouping of constraints makes detection, evaluation and resolution of order-centered conflicts difficult because the conflicting constraints are now dispersed across all resource-based subproblems.

Given the characteristics of each scheduling perspective, the control decision becomes one of how to partition the overall scheduling effort between perspectives such that the most important constraint conflicts can be directly addressed. One reasonable heuristic suggests that the most important resource-centered conflicts are likely to occur at those resources for which competition between orders is the greatest, i.e. scarce resources. These are more familiarly known as the bottleneck resources. Use of this heuristic results in a division of effort wherein a resource-based approach is used to schedule bottleneck resources and an order-based approach is used to schedule operations using non-bottleneck resources.

4.2. An Initial Multi-Perspective Scheduling System

To provide experimental justification for the claims put forth above regarding the use of multiple scheduling perspectives, an initial multi-perspective scheduling system was configured. A resource scheduling strategy based on the selective use of a set of dispatch scheduling heuristics was implemented, and the scheduling strategy of ISIS was adapted for use as the order scheduler. To simplify issues of coordination, the following, tightly controlled pattern of interaction between these two scheduling perspectives was imposed:

- The resource scheduler was first applied to a single, pre-specified bottleneck resource (a work area of the plant consisting of some number of machines).
- The order scheduler was then applied to work outward from this established portion of the shop schedule to complete the schedules for each individual order.

The performance of this system, designated OPIS 0, was then contrasted with that of ISIS and a dispatch system using the COVERT priority rule for minimizing tardiness cost (as formulated in [Vepsaleinen 84]). The latter system represents a well known and well regarded approach to job shop scheduling, and was included to provide a benchmark for the experimental study. We first discuss the approach to resource scheduling that was implemented. Next, the results obtained in the comparative analysis are summarized. Finally, the limitations of OPIS 0 implementation are identified. Further details of OPIS 0, and the design principles that guided its development, are contained in [Ow&Smith 86a, Ow&Smith 86b, Ow&Smith 86c, Smith 86b]. A complete account of the experiments may be found in [Ow 86].

4.2.1. Constructing Resource Schedules

The resource scheduling problem concerns the allocation of time on a designated resource to a set of competing operations involving different orders. In some cases, the designated resource is a single machine and the problem is solely one assigning execution times to the operations requiring the machine. More typically, however, the resource is a work area consisting of a set of machines, in which case the problem additionally involves an assignment of operations to specific machines. Quite often a given operation can only be performed on a subset of the machines that are contained in the area (i.e. machines in a given work area are functionally similar, but not necessarily identical).

There is an important observation concerning the role of resource scheduling in the context of generating a shop schedule that bears directly on the approach taken to resource scheduling. A global decision to solve a particular resource scheduling problem (as defined above) implies that contention for the designated resource is the crux of the problem (e.g. the resource appears to be a bottleneck). If contention wasn't a major problem, then there would be no resource scheduling problem and the global strategy would focus attention toward other, more critical scheduling decisions. Given this fact, assumptions can be made that limit the number of alternatives that must be considered during resource scheduling; in particular, we can safely assume that there is no need to consider the introduction of slack time between operations. This reduction in scope transforms the resource scheduling problem into a "dispatching" problem, and suggests an overall strategy for developing resource schedules. Scheduling decisions can be generated in an event-based fashion, with the scheduler repeatedly determining which operation to "dispatch" next to each machine in the work area under consideration.

Of course, the real issue of interest concerns how the dispatch decisions are to be made. The field of Operations Management (OM) has produced a large collection of dispatch scheduling heuristics. But, as we have already mentioned in Section 2, these heuristics are typically designed with respect to a particular objective (or set of objectives) and ignore all other relevant concerns. At the same time, there is no reason why a dispatch strategy must rigidly rely on a single scheduling heuristic (despite the fact that this is typically an assumption of OM research). If heuristics can be identified that cover the range of constraints that must be attended to, then knowledge of the importance relationships among constraints can be used in conjunction with knowledge of the current state of the solution to determine which heuristic to apply at any point.

The OPIS 0 resource scheduler is based on the above ideas. Resource schedules are developed in an iterative, event-based manner, with one or more scheduling decisions made on each iteration. The resource scheduling cycle is decomposed into three distinct phases which are carried out in

succession. A resource assignment phase is entered first, during which assignments of operations to machines are made. Next, a sequence development phase is entered to select a particular subset of assigned operations (at most one for each machine) to be scheduled on the current cycle. Finally, a resource allocation phase is entered wherein the selected operations are committed to. On any given cycle, the scheduler's attention is restricted to a subset of the most urgent operations that remain to be scheduled. This procedure for resource scheduling is made more precise in Figure 4-1.

---

1. [Initialization] - All unscheduled operations are sorted in increasing order of associated order due dates and tardy cost rates.

2. [Subproblem Selection] - If the sorted operation list is not empty, the first q operations (all if less than q in list) are selected to form the active operation set, Q. (Otherwise, stop.)

3. [Resource Assignment] - Each operation in Q is assigned to a specific machine in the designated bottleneck work-area. If the designated resource is a single machine, then this step is unnecessary.

4. [Subproblem Selection] - The set of machines to be scheduled on this iteration is determined. Only machines with operations assigned will be scheduled. Furthermore, if two assigned machines are substitutable for some operations in Q, only the one that is free earliest will be considered schedulable. Let the schedulable machines form the set M.

5. [Sequence Development] - For each m in M, the next operation to be scheduled on m is selected.

6. [Resource Allocation] - Reservations are made for the selected operations on the machine to which they have been assigned, and the selected operations are removed from the sorted operation list. (Go to step 2.)

Figure 4-1: The Resource Scheduling Cycle

Both the resource assignment and sequence development phases of the resource scheduling cycle make use of alternative scheduling heuristics. During resource assignment, the choice of heuristic is based on order slack, which is defined to be the time between the order's due date and its estimated completion time if it's operation is scheduled next (taking into account the earliest time that the order can arrive at the work area being scheduled). If the slack is long, e.g. 5 times the average processing time of the operation, then the heuristic applied is one that attempts to assign the operation to a machine with the proper setup. That is, tardiness is not expected to be a major problem and therefore the objective emphasized is setup minimization. Otherwise, a heuristic is applied that selects the

machine that results in the earliest completion time (emphasizing the tardiness cost minimization objective).

With respect to sequence development, the simple Earliest Due Date (EDD) rule works well when the proportion of tardy orders (or tardiness factor) is very low over a sequence of about 20 jobs, while the more complex Idle Time Rule [Ow 85] works well for higher tardiness factors. Therefore, the the Idle Time Rule is applied when the slack on orders are relatively short or negative. If the slack appears to be very long, either the EDD rule or a rule which attempts to select an operation that does not require any machine setup is applied. In our experiments, we have observed orders with shorter slack being scheduled early on in the resource scheduling process using the Idle Time Rule. Towards the later stages of the process, orders with longer lead times remain and these are scheduled using EDD or by product type to minimize setups.

4.2.2. A Comparative Analysis of Multi-Perspective and Single Perspective Scheduling

A scaled down model of the Winston Salem job shop was used to provide an environment for the experimental study. Six product types were included in the model, with associated process plans that utilized over 30 machines. Machines were functionally grouped into 11 work areas. The bottleneck area contained 7 machines. The orders to be scheduled were known in advance, and pre-determined due dates and tardy cost rates were used. For purposes of comparison, the schedules generated by each system were evaluated with respect to tardiness costs, work-in-process time, and the number of machine setups.

The comparative analysis was carried out over a total of 22 test problems. 20 of these test problems required 120 orders to be scheduled and the remaining 2 involved only 85 orders. Individual problems were generated by manipulating 4 parameters - the pattern of order releases (daily, weekly, exponentially rates), the number of orders released in each batch of releases, the product mix, and the setting of due dates. The set of problems was grouped into 18 categories, representing different shop conditions and load factors ranging from 70% to 120% of the capacity of the bottleneck area.

Figures 4-2 and 4-3 summarize the performance of each system with respect to tardiness cost and work-in-process time respectively. On each account, OPIS 0 was seen to outperform both ISIS and COVERT. Only four test problems were solved using ISIS largely because of the length of time taken to complete each task. Furthermore, it became clear from a detailed analysis of the ISIS schedules that the shortcomings predicted in using a purely order-based perspective were experienced. ISIS performed well with respect to minimizing work-in-process time as this depends primarily on an ability to resolve order-based conflicts. However, its performance with respect to tardiness cost suffered

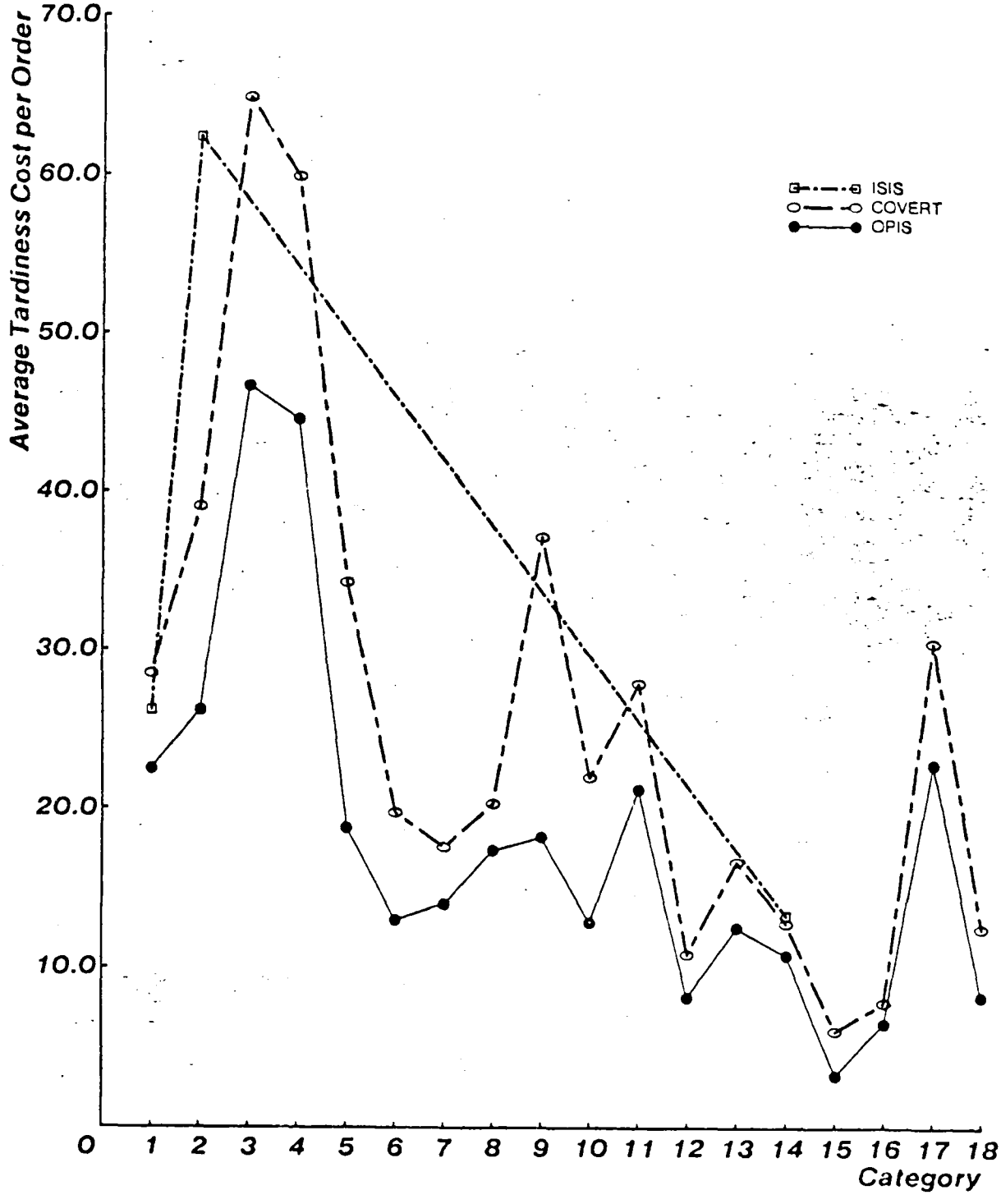


Figure 4-2: Average tardiness cost per order over different categories

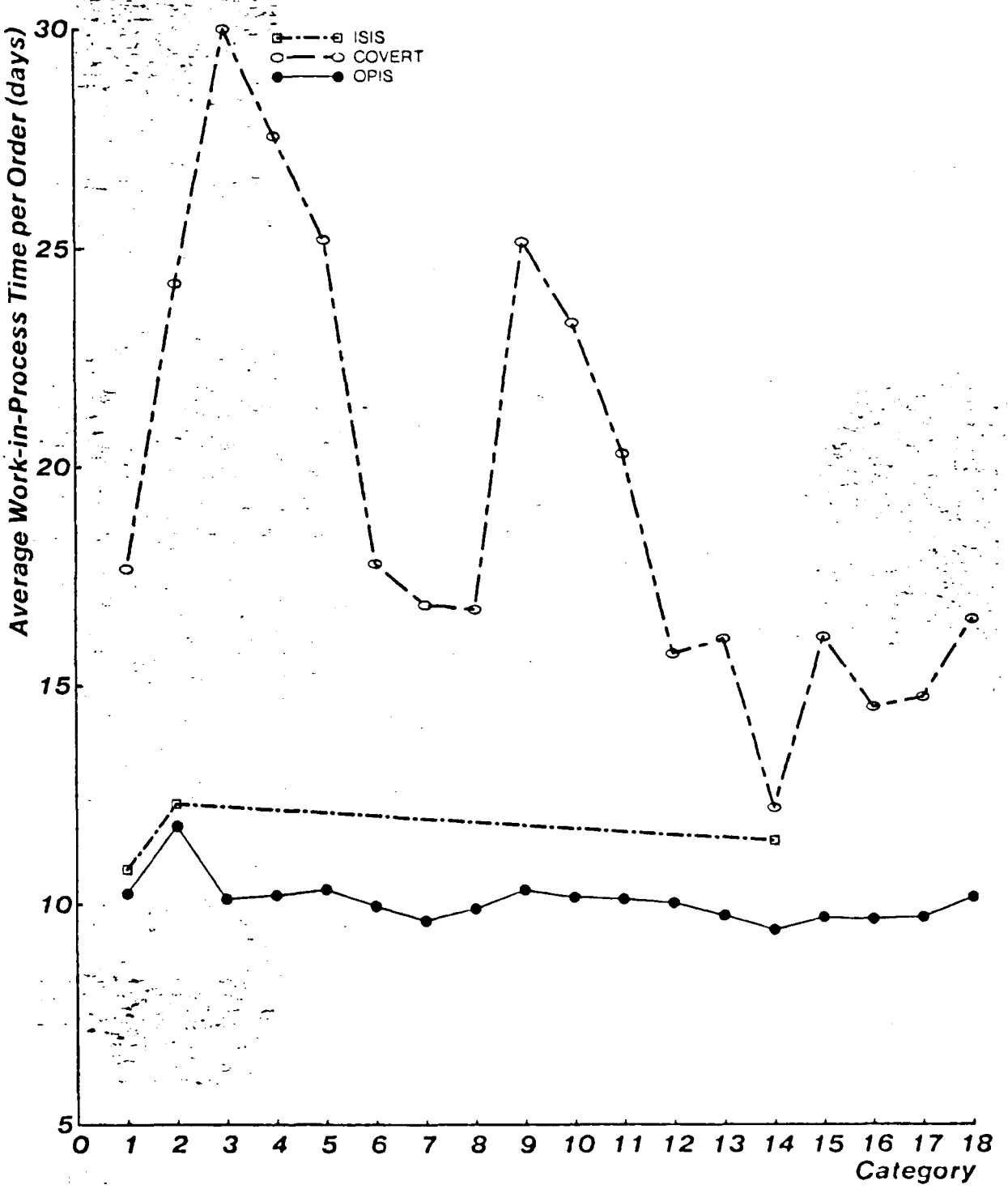
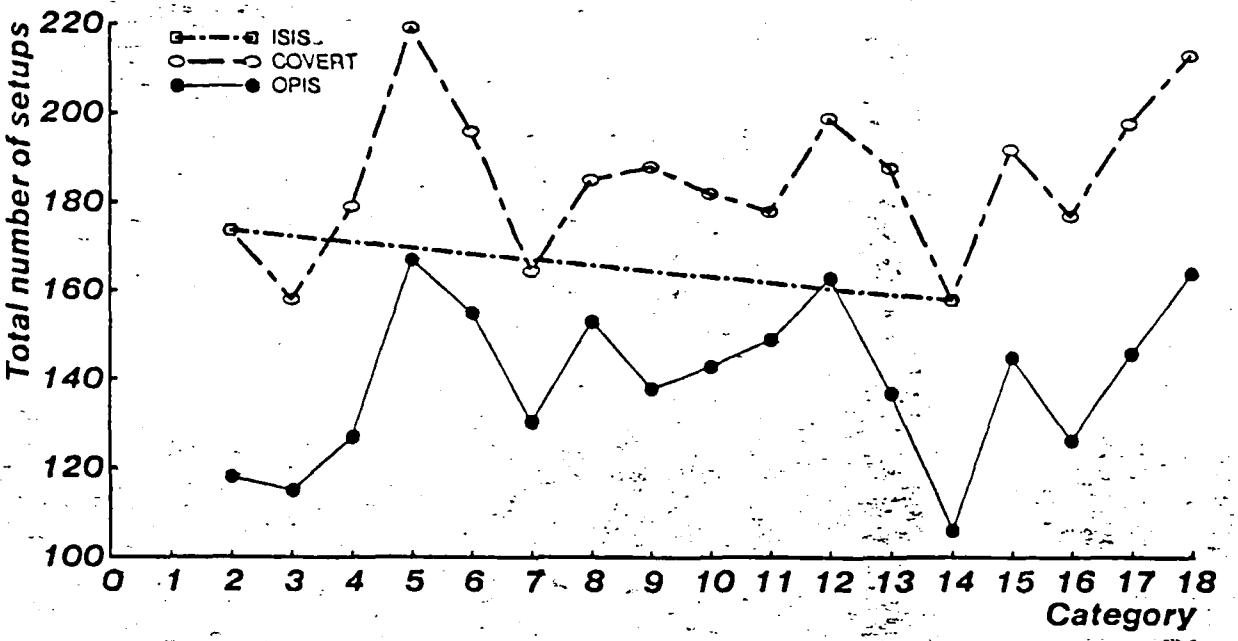
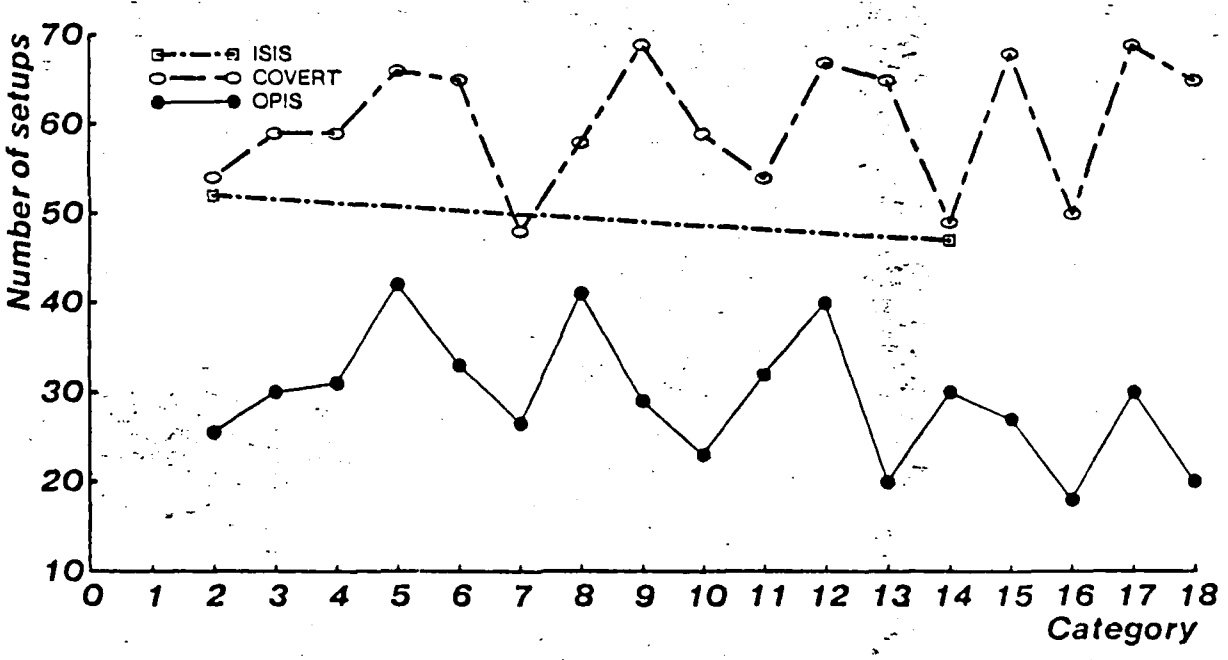


Figure 4-3: Average work-in-process time per order over different categories



a) Total number of setups for all resources



b) Total number of setups for bottleneck resources

Figure 4-4: Number of setups per shop schedule in each category

because of its inability to effectively handle resource-based conflicts. This fact is underscored by examining the number of setup changes in the schedules produced by OPIS 0 and ISIS (see Figure 4-4). The ISIS schedules contained close to twice as many setup changes for bottleneck machines as did the schedules generated by OPIS 0. The time required to perform these setup changes appeared to account for much of the discrepancy in tardiness cost performance. The variances in average tardy costs were also smaller for OPIS 0 than for ISIS and COVERT. Work-in-process time variances for OPIS 0 and ISIS were between 4 and 9 days while COVERT's ranged from 14 to 225 days because of the strictly local nature of the dispatch-based decision making.

4.2.3. Limitations of the Initial System Configuration

In implementing the OPIS 0 scheduler some rather severe limitations with respect to system flexibility were imposed. These limitations were felt to be justified, as our primary intent in constructing the system was to implement a particular multi-perspective scheduling strategy and provide some experimental evidence of the advantages of multi-perspective scheduling. In this section we examine the limitations of this implementation. This will serve to motivate the current OPIS scheduling architecture, which is described in Section 4.3.

A rather obvious restriction in OPIS 0 is its reliance on a static decomposition of the scheduling problem. A single pre-specified bottleneck resource is used to drive a fixed high level strategy for partitioning effort between the two scheduling perspectives. In an actual workshop, the situation is often much more complex. Several bottleneck resources may exist, either independently of one another or in a specific primary bottleneck/secondary bottleneck relationship. Furthermore, the bottleneck resource in the shop often "floats" over time, in which case specific resources need not be considered critical for the entire duration of the schedule. A priori specification of these more complex resource requirements is unreasonable, as many of the specific relationships emerge only during the scheduling activity (i.e. once some number of scheduling decisions have been made). An ability to dynamically predict "high contention" areas of the shop schedule is necessary to fully exploit the resource based scheduling perspective.

A second limitation of OPIS 0 concerns its strict assumption that the resource-based subproblem at the bottleneck resource completely dominates the order-based subproblems that involve non-bottleneck operations. The bottleneck schedule that is generated by the resource scheduler is guaranteed to be feasible (i.e. at least one conflict-free shop schedule is realizable), and the order scheduler is obliged to generate scheduling decisions that are consistent with the bottleneck schedule. This guarantee of feasibility is accomplished within the resource scheduler by actually building and maintaining a tentative (albeit simple) schedule for all resources required to perform

operations that must proceed the bottleneck operations. These tentative schedules are discarded once the bottleneck resource schedules have been finalized. There are two reasons why this is undesirable. First, the guarantee of feasibility is not exacted without the computational expense associated with generating the tentative schedules. More importantly, however, important concerns are likely to arise in subsequent subproblems which should force reconsideration of the existing bottleneck schedule (and subsequent compromise). The subproblem dominance assumptions made in OPIS 0 preclude this possibility.

A final limitation of OPIS 0, perhaps a more general statement of the limitation just described, is that it implements a schedule generation strategy and, as such, is insensitive to the dynamics of the shop floor. Unanticipated events (e.g. machine breakdowns, power failures) are typically commonplace on the shop floor, and continually introduce conflicts into the current shop schedule. Short of regenerating the entire shop schedule (which is obviously not often the desirable course of action), OPIS 0 has no capabilities for reacting to such events. At the same time, however, the attractiveness of multiple scheduling perspectives in responding to unanticipated events is fairly clear. There are some events that suggest a resource-centered perspective (e.g. a machine breakdown) while there are others that are more effectively addressed from an order-based perspective (e.g. a request for rework with respect to an in-process order).

4.3. A Scheduling Framework for Conflict-Directed Control

The limitations of OPIS 0 raised in the previous section all center around the need for greater system flexibility in approaching various scheduling tasks. In short, more dynamic and opportunistic control of problem decomposition and problem solving is required to fully address the scheduling requirements of actual factory environments. At the same time, OPIS 0 demonstrates the utility of a conflict-directed approach to structuring the search for a good schedule. In this case, a static assumption is made that the most critical constraint conflicts are those related to allocation of a prespecified bottleneck resource, and scheduling effort is focused there first. The OPIS 1 scheduling system (here after referred to simply as OPIS) generalizes from this example, defining a basic framework for conflict-directed control and using this framework to extend the OPIS 0 scheduling capabilities. In this section we focus on the OPIS control framework. The scheduling strategies currently implemented within this framework and the scheduling methods that these strategies rely on, are summarized in Section 4.4. The reader is referred to [Smith 86c] for further details.

A loosening of the reins on opportunistic problem decomposition and schedule generation introduces considerable additional complexity into the problem solving process. One must give up the assumption that individual subproblem results (or solution components) will be compatible and admit

the possibility that important local concerns will surface during the generation of specific solution components that lead to constraint violations when integrated with previously generated solution components. Thus, in addition to determining how the problem should be decomposed and in which order various components of the schedule should be generated, the scheduler must be capable of monitoring progress made toward a final schedule, recognizing and characterizing conflicts in the schedule as they arise, and using these characterizations to initiate appropriate schedule revision activities.[10] Note, however, that these capabilities also provide the necessary machinery for responding to unanticipated external events. This process differs only in the fact conflicts are introduced into the predictive schedule through indications that the status of the factory has changed. The OPIS control architecture described below provides these capabilities, and, in doing so, provides a framework that merges the activities of predictive schedule generation/expansion and reactive schedule maintenance.

4.3.1. Overview

Figure 4-5 depicts the top level structure of OPIS, and identifies the major components of the current system architecture. The organization is a variation of the HEARSAY-II blackboard style architecture [Erman 80], and similarly assumes a system organization comprised of a number of knowledge sources (KSs) that extend and revise a global set of one or more hypotheses. In this case, the KSs implement alternative scheduling methods and the hypotheses being manipulated are candidate shop schedules. For simplicity in the following discussion we will assume that only a single shop schedule is being manipulated.[11]

Within this architecture, a designated KS called the manager assumes responsibility for planning and coordinating the scheduling effort. Scheduling proceeds via the formulation and initiation of *scheduling tasks*. Each scheduling task requests a particular analysis, extension or revision relative to the current shop schedule (e.g. generate a schedule for work area *wa-1*, revise the schedule for order *ord-1*, analyze the capacity of the shop), and designates a specific scheduling KS to carry out the

---

[10] Given the granularity of the solution components that are being synthesized (e.g. a schedule for a specific work area) and the high degree of interdependency among the decisions that comprise these solution components (due to the temporal and resource constraints on the problem), systematic backtracking procedures are of little use in resolving conflicts. The system's approach to revision must be driven by characteristics of the conflict (or conflicts) at hand.

[11] Note that this does not mean that there is no search taking place; but rather that exploration of alternative scheduling decisions is confined to the local subproblems addressed by individual KSs. For example, in constructing a given order's schedule the order scheduler will conduct a search before committing to a particular set of scheduling decisions. Once this commitment has been made, however, it becomes part of a single evolving shop schedule. This "single shop schedule" assumption is necessary in the context of schedule generation to keep the problem computationally tractable. Recall from Section 3.4 that this assumption was built into the ISIS search architecture. The provision for maintaining multiple shop schedules is included in the OPIS architecture primarily for reacting to external events. Here, the problem is smaller in scope, and it is quite feasible to consider alternative schedule revision strategies.

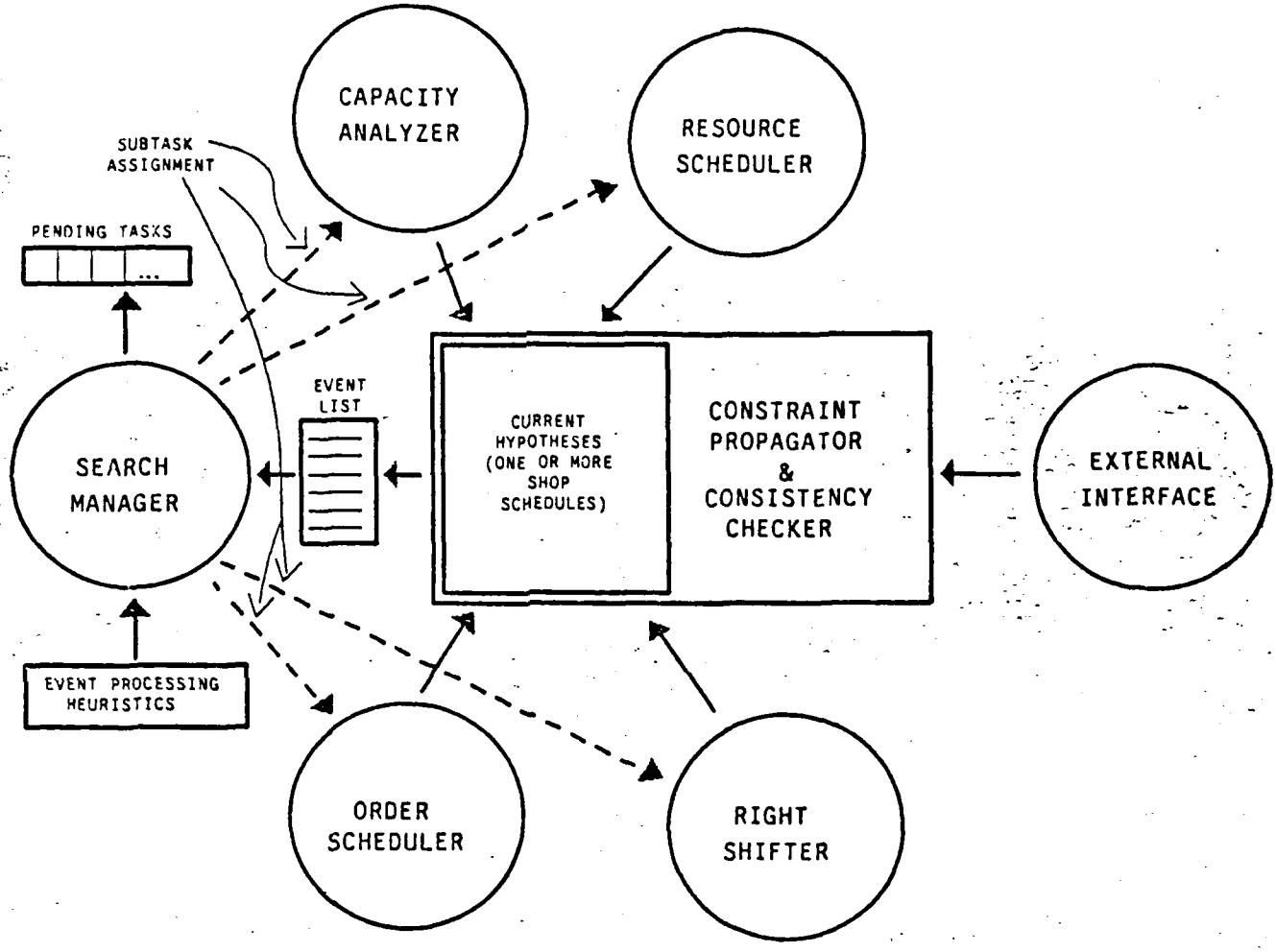


Figure 4-5: Current OPIS Architecture: Top Level

task. The manager's queue of *pending subtasks* constitutes its current plan for solving the scheduling problem at hand. The execution of a scheduling tasks by scheduling KSs yields changes to current shop schedule. These changes are integrated into the current hypothesis by the *schedule management subsystem*, which exploits the temporal restrictions and capacity limitations specified in the factory model to determine the additional constraints imposed on the schedule by each new scheduling decision. This provides an accurate characterization of the current state of the evolving solution and a straightforward basis for detecting conflicts. The manager is informed of the results of

KS execution through the posting of *control events*, which summarize those aspects of KS execution that are of importance from a control perspective. Events may highlight important characteristics of the current solution constraints (e.g. resource contention is likely to be high at a particular resource), indicate that progress has been made toward a final solution (e.g. another component of the shop schedule has been generated), or identify specific problems that are present the solution (e.g. constraint violations). The manager responds to posted events through application of a set of *event processing heuristics*. This results in the formulation of new scheduling tasks, and the queue of pending subtasks is updated accordingly. Thus, the manager implements an event-driven control regime, continually revising its "scheduling plan" as the results of KS execution become known.

At the core of this framework for control are two key notions:

1. The use of a centralized schedule management component as a means of communicating constraints among subproblems and recognizing constraint violations; and

2. An event-based framework for representing and structuring search control knowledge.

These two notions are elaborated in the following subsections.

4.3.2. Schedule Management

The OPIS scheduling architecture makes no commitment as to the order in which individual scheduling decisions will be made. Rather, the architecture assumes that characteristics of the problem at hand, in particular analyses of the constraint conflicts that must be resolved, will be used to dynamically prioritize the scheduling decisions that have to be made. Thus, for example, schedule generation might proceed by constructing schedules for resources where contention is likely to be high and then considering the scheduling decisions that involve the allocation of less critical resources. Similarly, indication that a machine will be down for the next week might lead to some amount of rescheduling in the work area containing the failed machine, followed by revision of any scheduling decisions in other parts of the schedule that are affected by this rescheduling. Essential to this opportunistic approach to scheduling is an ability to maintain an accurate characterization of the current state of the schedule. Both problem decomposition (the formulation of appropriate scheduling tasks) and subproblem solution (the application of specific scheduling KSs) require knowledge of the constraints that are currently imposed on the schedule (both by the current factory state and the scheduling decisions that have already been made).

This support is provided within OPIS by incrementally maintaining an explicit representation of (1) the current temporal constraints on each manufacturing operation that must be scheduled, and (2) the current availability of each resource. Thus, an operation description delineates, at any point, the

set of allocation decisions that are compatible with the constraints imposed by the external world and any other scheduling decisions that have already been made. This is illustrated by the partial description in Figure 4-6, which states that operation **ord2-P1-root-grinding** must be scheduled in the **208-rooting-area** sometime between 10:30 and 17:00 on August 18th to remain consistent with the current solution. In this case, the end time constraint is a consequence of the scheduling decision that was made concerning the downstream operation **ord2-P1-milling-operation**, and the start time constraint is a consequence of both the order's release date and unavailability of the 208 rooting machines (perhaps due to their prior allocation to other operations). Descriptions of resource availability, which are associated with the resources themselves, characterize intervals of time during which the resource may still be allocated and how much available capacity remains (in the case of aggregate resources). These representations of current time and capacity constraints are maintained at different levels of aggregation to enable scheduling at various levels of precision.

```
{{ord2-P1-root-grinding
    {INSTANCE: P1-root-grinding-operation
        ORDER: order2
        RESOURCE-REQUIREMENTS: 208-rooting-area
        NEXT-OPERATION: ord2-P1-milling-operation
        STATUS: unscheduled
        TIME-BOUND-INTERVAL: {{INSTANCE: calendar-time-interval
                              START-TIME: Aug 18 10:30
                                  origins: (order-release-date order2)
                                           (capacity-restriction 208-rooting-area)
                              END-TIME: Aug 18 17:00
                                  origins: (scheduled-start-time ord2-P1-milling-operation)
                              }} } }}
```

Figure 4-6: An unscheduled operation with time bound constraints

These constraint representations are maintained by a set of propagation processes collectively referred to as the schedule management subsystem. The propagation processes are driven by the temporal restrictions (e.g. operation precedences, operation durations) and resource requirements specified in the factory model, and are triggered whenever changes to the schedule are made. When operations are scheduled, for example, their descriptions are updated to reflect the chosen resources and intervals of execution, and constraints are propagated to both related operations and the resources that have been allocated. Changes to the schedule may be more complex than the addition or retraction of individual scheduling decisions. For example, indication that an order must be

reworked requires the addition of the necessary rework operations to the production plan before time bound propagation can be carried out. Specific "schedule update" processes are defined for each type of change to the schedule that might be encountered.

As mentioned previously, this explicit representation of the current state of the schedule serves two purposes within the OPIS scheduling architecture. The first is one of constraint communication, making all current constraints apparent to scheduling KSs during generation or revision of specific components of the overall schedule. The second is in providing a basis for detecting problems in the schedule. The representation we have described enables straightforward detection of three basic types of constraint conflicts: *time* conflicts (corresponding to precedence violations) *capacity* conflicts (corresponding to resource availability violations), and *time vs capacity* conflicts (corresponding to situations where scheduling decisions do not exist that mutually satisfy current temporal restrictions and resource availability constraints). In OPIS, detection of these types of conflicts is coupled with the constraint propagation processes, and each conflict detected is posted with the manager (see below). Complete details of the constraint propagation processes and the detection of conflicts may be found in [LePape&Smith 86].

4.3.3. Event-Based Control

The OPIS manager formulates, extends, and revises its current scheduling plan (i.e. its queue of pending subtasks) in response to posted *control events*. Control events represent those consequences of internally initiated scheduling actions (i.e. KS execution) and externally initiated schedule updates that are relevant to control decisions. They are generated and posted as a result of either activity. Events provide an abstract view of the current state of the schedule, and contain all the information necessary for the manager to determine how to proceed. Events also provide a means for organizing the system's control knowledge. *Event processing heuristics*, which map occurrences of particular events to appropriate sequences of scheduling tasks, and knowledge relating to event importance, which is used in ordering the queue of pending subtasks, are directly attached to the prototype description of each event type and therefore directly accessible to the manager in responding to specific events. These ideas are illustrated in Figure 4-7.

The set of control events defined within the OPIS architecture are categorized into three general classes:

- *conflicts* - Conflict events are used to characterize inconsistent sets of scheduling decisions that have been detected in the schedule. Such events involve the violation of non-negotiable constraints and are precisely those that are detectable by the schedule management subsystem. Reaction in this case is mandatory as the current schedule is infeasible.

```
{{precedence-violation
    {IS-A: elementary-conflict
        CONFLICTING-COMMITMENTS:
        HYPOTHESIS:
        MAGNITUDE:
        INTRODUCED-BY:
        EVENT-TYPE-IMPORTANCE: 4
        EVENT-PROCESSING-HEURISTICS: pv-heur1 pv-heur2
        process-event: process-event
        calculate-overall-significance: calc-pv-significance
        verify-control-state: pv-state-check } }}
```

Figure 4-7: The **precedence-violation** event prototype

- *compromises* - Compromise events are produced as the result of analysis tasks and concern the violation of preference constraints. Two event subtypes are distinguished here: *unsatisfactory compromises*, which identify preference concerns that have been unacceptably relaxed, and *predicted compromises*, which designate areas in the schedule where it appears that it will be necessary to compromise preferences. Unsatisfactory compromise events may or may not be reacted to, depending on the manager's perception of the opportunities for improvement. Predicted compromises provide a basis for prioritizing the set of scheduling decisions that must be made.

- *hypothesis-modifications* - Hypothesis modification events simply indicate changes that have been made to the current schedule. They are posted as a result of either KS execution or factory status updates. In the former context, hypothesis modification events provide a basis for stringing together specific sets of subtask creation heuristics to implement particular schedule development strategies.

Upon initial invocation and at the end of each top-level problem solving step, the OPIS manager responds to the currently posted set of events. This activity proceeds in two steps:

1. *event aggregation*, during which the most appropriate set of events to consider is determined, and
2. *event processing*, during which the manager's current scheduling plan is revised in response to this determined set of events.

During the event aggregation step, any "related" events among those that have been posted are combined into aggregate events. Because of the fact that KS execution and external status updates may result in a considerable amount of change to the current solution (e.g. the resource scheduler may make decisions for all orders that pass through a given work area), several constraint conflicts can be introduced during a single top-level problem solving step. As we have seen, each conflict is

detected and reported individually during constraint propagation. It is often the case that these elementary events are related in some manner (e.g. they involve different operations of the same order, they involve different operations requiring the same resource, etc.), and would be better addressed by the system simultaneously. Thus, the notion of an aggregate event is introduced, and aggregate event types are defined on the basis of such relationships. *Event aggregation heuristics* are associated with these descriptions to specify the precise circumstances under which two or more posted events should be transformed into an aggregate event of a given type.

During the event processing step, the event processing heuristics associated with each currently posted event (including any aggregate events generated during the event aggregation step) are applied to determine how the system should proceed. These condition/action rules examine characteristics of the event being processed and specify extensions and revisions to the manager's current queue of pending subtasks. These changes involve some combination of the following primitive actions: the creation of new subtasks to perform, a reordering of existing tasks in the queue, and the elimination of existing tasks. Once all events have been processed, the queue of pending subtasks is updated and the highest priority pending subtask is initiated. Subtask prioritization is a function of the significance of the triggering event type (e.g. tasks resulting from conflict events are generally considered more important than those resulting from hypothesis modification events), characteristics of the *triggering* event (e.g. the magnitude of the conflict reported), and characteristics of the task itself (e.g. its dependencies with respect to other pending subtasks).

4.4. Scheduling Knowledge Sources

Four scheduling KSs have been implemented within the architectural framework described above. The order scheduler of OPIS 0 (essentially the detailed resource analysis and resource assignment levels of ISIS - see Section 3.4.2) has been revised to operate with the propagated time bound constraints and provide the reactive capability described in Section 3.4.4. A schedule revision capability has also been added to the resource scheduler of OPIS 0. The strategy implemented attempts to retract only as many scheduling decisions as necessary to produce a new, conflict-free schedule for the designated resource. This is accomplished by assuming no schedule forward in time from the point of the current problem (e.g. order contention due to insufficient capacity), and invoking the schedule generation strategy (see Section 4.2.1). However, after each new set of allocation decisions is made (i.e. after each resource scheduling cycle), an analysis is performed to see if the new schedule can be consistently synthesized with the fragment of the old schedule consisting of the operations that have yet to be placed in the new schedule. Thus, both detailed scheduler KSs may be used in either a schedule generation/refinement or schedule revision mode.

Two additional KSs have also been added. The first, referred to as the capacity analyzer[12], implements a "shop level" scheduling perspective. It provides a basis for dynamic problem decomposition by generating predictions of likely areas of high resource contention. In contrast to the detailed schedulers, the capacity analyzer operates with aggregate descriptions of resources, operations and resource allocation decisions. It constructs a predictive shop schedule that satisfies the time bound constraints posted with each aggregated operation, using a general line balancing heuristic. The demand for capacity reflected by this schedule is then compared with the actual capacity of the required aggregate resources and likely bottlenecks are predicted. The second KS added to the configuration implements a simple schedule-shifting strategy and is employed to resolve minor inconsistencies that might arise.

The manager's current body of control heuristics generalizes the OPIS 0 schedule generation strategy to one where effort is initially focused on scheduling any number of predicted bottlenecks (as dynamically determined by the capacity analyzer). The schedule is then completed on an order by order basis, and contingencies are included for revising decisions made by the resource scheduler in response to inconsistencies that arise later on in the search. Heuristics are also in place for reactively revising the current schedule as status updates are received from the factory floor. These heuristics define strategies for responding to machine failures, operation delays, rework requests, and the receipt of new orders.

5. Conclusions

The work described in this report was undertaken with the hypothesis that the use of constraint knowledge is central to obtaining a good solution to the job shop scheduling problem, and has sought to develop a scheduling methodology that is driven by such knowledge. Several key elements of a theory of constraint-directed job shop scheduling have emerged from this work:

- Analysis of specific job shop environments has resulted in an identification and categorization of the various types of constraints that influence scheduling decisions (Section 2). This categorization broadly delineates the knowledge requirements of a constraint-directed scheduling system.

- A representational framework for modeling all aspects of the manufacturing enterprise has been developed (Section 3.1). The framework ascribes a semantics to general concepts of activities, states, objects, causality, and time, which is then refined to provide primitives for modeling the production environment and its constraints.

---

[12] Despite the name, this KS bears no relationship to the capacity analysis level of the ISIS search architecture. In fact, the ISIS capacity analysis level has been subsumed by the propagation techniques of the schedule management component and removed from the order scheduler.

- A constraint representation that extends predicate constraint specification techniques to enable the expression of preference constraints has been designed (Section 3.2). This defines the notion of relaxable constraints (i.e. constraints that may be satisfied to varying degrees, depending on the specific situations of constraint conflict that arise during scheduling). The representation makes alternative choices explicit, and formalizes the knowledge necessary to intelligently relax (or compromise) specific constraints in conflicting situations.

- A methodology for constraint resolution (i.e. determination of precisely which constraints are relevant to a given scheduling decision) has been developed, based on (1) model semantics concerning placement of constraints in the model (Section 3.3.1), and (2) techniques for propagating those constraints that are created dynamically during scheduling through the model (Section 4.3.2).

- A dynamic schedule evaluation scheme, based on knowledge defined in the constraint representation and intuitively reflecting how well the set of scheduling decisions under evaluation satisfies the relevant objectives and preferences, has been developed (Section 3.3.1). This evaluation scheme is useful in two contexts: it provides a basis for comparing alternative sets of scheduling decisions that are generated during the search for a good schedule (Section 3.4.2), and it provides a means for assessing the quality of user imposed scheduling decisions (Section 3.5).

- Both generative and analytic techniques for intelligently compromising among conflicting constraints have been developed (Sections 3.3.2, 4.1). Generative (or search-based) relaxation utilizes the alternatives possible with respect to one constraint (e.g. the resources capable of performing a given operation) to generate a set of alternative decisions, each of which will variably relax other relevant constraints (e.g. due date constraint, resource preferences). Analytic (or rule-based) relaxation exploits knowledge relating to the importance of and interdependencies between various constraints to either restrict or redirect the search to specific areas of the solution space. This focus results in an emphasis on specific constraints and a de-emphasis on others.

- Both hierarchical and opportunistic search architectures have been investigated as frameworks for controlling the combinatorics of the search for a good schedule. Hierarchical techniques have been defined that enable a "staged" introduction of constraints via multiple levels of analysis, where the results of each level provide insight regarding the solution at lower levels (Section 3.4). Opportunistic techniques have been defined that exploit knowledge about constraint conflicts to dynamically structure the search (Section 4.3).

- Several approaches to integrating Operations Management (OM) methods within a constraint-based scheduling framework have been demonstrated. Specific dispatch heuristics can be injected in the formulation of various due date constraints (Section 3.2). Techniques for selectively applying different dispatch heuristics, based on characteristics of the current state of the solution and the relationships among the constraints that are emphasized by each heuristic, have also been demonstrated (Section 4.2.1).

Several prototype scheduling systems based on these concepts and techniques have been constructed, and comparative analyses of the performance characteristics of these systems and a

well regarded traditional scheduling method have been conducted in the context of an actual manufacturing facility (Sections 3.6, 4.2.2). The results obtained provide experimental evidence of both the viability and the potential of constraint-directed scheduling.

The results of this research by no means constitute a final solution to the job shop scheduling problem. There are many difficult issues related to understanding the longer term implications of short term reactive adjustments to the schedule, for example, that we have only just begun to address. At the same time, this research has introduced, for the first time, a scheduling methodology that explicitly addresses the diversity of constraints that actually influence factory operation. While a complete theory of constraint-directed scheduling has not yet been achieved (i.e. it is not yet possible to base *all* system decision-making on a declarative specification of constraint knowledge), concepts and techniques have been demonstrated that offer opportunities for substantial improvement in factory performance. Furthermore, since the job shop scheduling problem has much in common with other types of logistics support problems, we feel that the results of this research are applicable to a much broader class of scheduling problems.

References

[Allen 81] Allen, J.F.
A General Model of Action and Time.
Technical Report 97, University of Rochester, November, 1981.

[Allen&Koomen 83]
Allen, J.F. and J.A. Koomen.
Planning Using a Temporal World Model.
In *Proceedings Eighth International Joint Conference on AI.* Karlsruhe, West Germany, August, 1983.

[Daniel 84] Daniel, L.
Planning and Operations Research.
In O'Shea, T., and M. Eisenstadt (editors), *Artificial Intelligence: Tools, Techniques, and Applications.* Harper and Row, 1984.

[Engleman 80] Engleman, C., E. Scarl and C. Berg.
Interactive Frame Instantiation.
In *Proceedings of the 1st National Conference on Artificial Intelligence*, pages 184-186. American Association of Artificial Intelligence, 1980.

[Erman 80] Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R.
The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty.
Computing Surveys 12(2), June, 1980.

[Fox 82] Fox, M. S., B.P. Allen and G.A. Strohm.
Job Shop Scheduling: An Investigation in Constraint-Directed Reasoning.
In *Proceedings of the 2nd National Conference on Artificial Intelligence*, pages 155-158. American Association of Artificial Intelligence, August, 1982.

[Fox 83a] Fox, M.S.
Constraint-Directed Search: A Case Study of Job Shop Scheduling.
PhD thesis, Computer Science Department, Carnegie-Mellon University, 1983.

[Fox 83b] Fox, M.S., Smith, S.F., Allen, B.P., Strohm, G.A., and Wimberly, F.C.
ISIS: A Constraint-Directed Reasoning Approach to Job Shop Scheduling.
In *Proceedings IEEE Conference on Trends and Applications 83*. Gaithersberg, Maryland, May, 1983.

[Fox 85] Fox, M.S.
Knowledge Representation for Decision Support.
In Methlie, L.B. and Sprague, R.H. (editors), *Knowledge Representation for Decision Support Systems*. North-Holland, Amsterdam, 1985.

[Fox 86] Fox, M.S.
Observations on the Role of Constraints in Problem Solving.
In *Proceedings 6th Annual Conference of the Canadian Society for Computational Studies of Intelligence*. Montreal, Canada, July, 1986.

[Fox&Smith 84a] Fox, M.S., and S.F. Smith.
ISIS: A Knowledge-Based System for Factory Scheduling.
Expert Systems 1(1):25-49, July, 1984.

[Fox&Smith 84b] Fox, M.S. and S.F. Smith.
The Role of Intelligent Reactive Processing in Production Management.
In *Proceedings 13th Annual CAMI Technical Conference*. Clearwater, Florida, November, 1984.

[Fukumori 80] Fukumori, K.
Fundamental Scheme for Train Scheduling (Application of Range-Constriction Search).
A.I. Memo 596, Massachusetts Institute of Technology, AI Lab, September, 1980.

[Garey&Johnson 79]
Garey, M.R. and D.S. Johnson.
Computers and Intractability.
W.H. Freeman and Company, San Francisco, 1979.

[Goldstein&Roberts 77]
Goldstein, I.P., and R.B. Roberts.
NUDGE: A Knowledge-Based Scheduling Program.
In *Proceedings 5th IJCAI*, pages 257-263. 1977.

[LePape&Smith 86]
LePape, C. and S.F. Smith.
Management of Temporal Constraints for Factory Scheduling.
Technical Report, Robotics Institute, Carnegie Mellon University, forthcoming, 1986.

[McDermott 82] McDermott, D.
A Temporal Logic for Reasoning About Processes and Plans.
Cognitive Science 6:101-155, 1982.

[Miller 83] Miller, D.
Scheduling Heuristics for Problem Solvers.
Research Report 264, Yale University, Dept. of Computer Science, April, 1983.

[Ow 85] Ow, P.S.
Focused Scheduling in Proportionate Flowshops.
Management Science 31(7), July, 1985.

[Ow 86] Ow, P.S.
Experiments in Knowledge-Based Scheduling.
Technical Report, CMU Robotics Institute Technical Report, forthcoming, 1986.

[Ow&Smith 86a] Ow, P.S. and S.F. Smith.
Toward an Opportunistic Scheduling System.
In *Proceedings 19th Hawaii International Conference on System Sciences.*
January, 1986.

[Ow&Smith 86b] Ow, P.S. and S.F. Smith.
Two Design Principles for Knowledge-Based Systems.
Decision Sciences , to appear, 1986.

[Ow&Smith 86c] Ow, P.S. and S.F. Smith.
Viewing Scheduling as an Opportunistic Problem Solving Process.
Technical Report, Robotics Institute, Carnegie-Mellon University, forthcoming, 1986.

[Sathi 85] Sathi, A., M.S. Fox, and M. Greenberg.
Representation of Activity Knowledge for Project Management.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-7(5), September, 1985.

[Smith 83] Smith, S.F.
Exploiting Temporal Knowledge to Organize Constraints.
Technical Report CMU-RI-TR-83-12, CMU Robotics Institute, July, 1983.

[Smith 86a] Smith, S.F., M.S. Fox, and P.S. Ow.
Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems.
AI Magazine 7(4), Fall, 1986.

[Smith 86b] Smith, S.F., P.S. Ow, C. LePape, B. McLaren, and N. Muscettola.
Integrating Multiple Scheduling Perspectives to Generate Detailed Production Plans.
In *Proceedings SME Conference on AI in Manufacturing*. Long Beach, CA, September, 1986.

[Smith 86c] Smith, S.F., P.S. Ow, C. LePape, and N. Muscettola.
Reactive Management of Factory Schedules.
Technical Report, CMU Robotics Institute Technical Report, in preparation, 1986.

[Smith&Ow 85] Smith, S.F. and P.S. Ow.
The Use of Multiple Problem Decompositions in Time-Constrained Planning Tasks.
In *Proceedings 9th International Joint Conference on Artificial Intelligence*. Los Angeles, California, August, 1985.

[Stefik 81] Stefik, M.
Planning With Constraints (MOLGEN Part 1).
Artificial Intelligence 16:111-140, 1981.

[Sussman&Steele 80]
Sussman, G.J. and G.L. Steele Jr.
CONSTRAINTS - A Language for Expressing Almost-Hierarchical Descriptions.
Artificial Intelligence 14:1-40, 1980.

[Vepsaleinen 84] Ari Vepsaleinen.
State Dependent Priority Rules for Scheduling.
PhD thesis, Carnegie-Mellon University, April, 1984.

[Vere 81] Vere, S.
Planning in Time: Windows and Durations for Activities and Goals.
Technical Report, Jet Propulsion Laboratory, 1981.

[Waltz 75] Waltz, D.
Understanding Line Drawings of Scenes with Shadows.
In P.H. Winston (editor), *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.

[Wright&Fox 83] Wright, Mark and Mark S. Fox.
SRL/1.5 User Manual.
Technical Report, The Robotics Institute, Carnegie-Mellon University, December, 1983.